

左　军　主编

清华大学出版社

Android 程序设计经典教程

左军　主编

清华大学出版社
北　京

内 容 简 介

本书从初学者的角度出发，通过通俗易懂的语言、丰富多彩的实例介绍了 Android 程序开发的各方面技术。本书在介绍 Android 技术的同时，提供一些经典案例，通过经典案例让读者快速掌握 Android 技术。本书除了纸质内容之外，还提供了所有案例的源程序代码。全书共 10 章，主要包括 Android 初识、布局、控件、视图、动画、对话框、菜单、程序组件、通信以及开发等内容。

本书适合 Android 入门级开发人员，初、中级程序员，特别适合于程序开发人员作为 Android 开发的参考书。

图书在版编目(CIP)数据

Android 程序设计经典教程/左军主编. --北京：清华大学出版社，2015
ISBN 978-7-302-39255-2

Ⅰ.①A…　Ⅱ.①左…　Ⅲ.①移动终端－应用程序－程序设计－高等学校－教材　Ⅳ.①TN929.53

中国版本图书馆 CIP 数据核字(2015)第 024304 号

责任编辑：魏江江　赵晓宁
封面设计：
责任校对：李建庄
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**27.5　　**字　　数：**686 千字
版　　次：2015 年 4 月第 1 版　　**印　　次：**2015 年 4 月第1 次印刷
印　　数：1～　000
定　　价：　.00 元

产品编号：062457-01



1. Android 介绍

手机是人们工作和生活中不可缺少的产品，而智能手机则极大地扩展了手机的功能。智能手机越来越多地为用户所接受。从近期表现来看，手机操作系统已成为智能手机厂商间的竞争重点。市场上形成了 Android 阵营、Symbian 阵营与苹果自有 Mac 操作系统阵营的三足对抗之势。在这三者中，最具有发展潜力的是 Android 阵营。

Android 早期由"Anandroid 机器人 Android 之父"之称的 Andy Rubin 创建，Google 公司于 2005 年并购了成立仅 22 个月的高科技企业 Android，展开了短信、手机检索、定位等业务，基于 Linux 的通用平台进行了开发。

虽然 Android 面世时间不长，但 Android 对传统的手机平台构成了极大的威胁。Android 在国内的前景十分广阔，首先是有成熟的消费者，Android 社区十分红火，这些社区为 Android 在中国的普及做了很好的推广作用。国内厂商和运营商也纷纷加入了 Android 阵营，包括中国移动、中国联通、中兴通讯、华为通讯、联想等大企业，同时不仅局限于手机，国内厂家也陆续推出了采用 Android 系统的 MID 产品。

随着移动互联网的到来和迅猛发展，移动互联网开发人员的需求也是与日俱增。目前中国拥有世界上最大的手机用户群，再加上 3G 的推出对整个行业的巨大推动作用，全世界所有大中小型手机制造商几乎都在招聘 Android 工程师。然而每天有超过 16 万台的 Android 设备出货，以后将超越 iPhone 成为智能手机平台的旗舰。

Android 具有如下 5 大优势：

(1) 开放性。

在优势方面，Android 平台首先就是其开发性，开发的平台允许任何移动终端厂商加入到 Android 联盟中来。显著的开放性可以使其拥有更多的开发者，随着用户和应用的日益丰富，一个崭新的平台也将很快走向成熟。

开发性对于 Android 的发展而言，有利于积累人气，这里的人气包括消费者和厂商，而对于消费者来讲，随大的受益正是丰富的软件资源。开放的平台也会带来更大竞争，如此一来，消费者将可以用更低的价位购得心仪的手机。

(2) 挣脱运营商的束缚。

在过去很长的一段时间，特别是在欧美地区，手机应用往往受到运营商制约，使用什么功能接入什么网络，几乎都受到运营商的控制。从去年 iPhone 上市，用户可以更加方便地连接网络，运营商的制约减少。随着 EDGE、HSDPA 这些 2G 至 3G 移动网络的逐步过渡和提升，手机随意接入网络已不是运营商口中的笑谈了。

(3) 丰富的硬件选择。

这点还是与 Android 平台的开放性相关，由于 Android 的开放性，众多的厂商会推出千奇

百怪、功能特色各具的多种产品。功能上的差异和特色，却不会影响数据同步、甚至软件的兼容，就像从诺基亚 Symbian 风格手机一下改用苹果 iPhone，同时还可将 Symbian 中优秀的软件带到 iPhone 上使用、联系人等资料更是可以方便地转移。

(4) 不受任何限制的开发商。

Android 平台提供给第三方开发商一个十分宽泛、自由的环境，不会受到各种条条框框的阻挠，可想而知，会有多少新颖别致的软件会诞生。但也有其两面性，血腥、暴力等方面的程序和游戏如何控制正是留给 Android 难题之一。

(5) 无缝结合的 Google 应用。

如今叱咤互联网的 Google 公司已经走过 10 年了，从搜索巨人到全面的互联网渗透，Google 服务如地图、邮件、搜索等已经成为连接用户和互联网的重要纽带，而 Android 平台手机将无缝结合这些优秀的 Google 服务。

2. 本书特点

鉴于 Android 作为新的平台、新技术，为了帮助众多开发人员和爱好者进入 Android 开发领域，并提高程序开发水平，编写了本书。本书具有如下写作特点：

(1) 内容全面丰富：对于刚接触 Android 的人员，本书首先对 Android 系统的历史以及架构做了一个细致的介绍，对每一个知识点都配有相应的图片及说明。

(2) 通俗易懂：本书条理清晰、语言简洁，可帮助读者快速掌握每个知识点；每个部分既相互连贯又自成一体，使读者可以按照本书编排的章节顺序进行学习，也可以根据自己的需求对某一章节进行针对性的学习。

(3) 实用性强：本书彻底弃除枯燥的理论和简单的操作，注重实用性和可操作性，将 Android 网络开发技术的理论融合到实际的操作环境中，使用户在掌握相关操作技能的同时，还能够学习到相应的开发知识。

(4) 实例经典：书中的实例应用全面，涵盖了 Android 所能触及的领域。实例代码翔实、规范工整，且代码注释得当。

3. 本书内容

本书共 10 章，其主要介绍内容如下：

第 1 章　介绍了 Android 初识，主要包括 Android 的发展史、Android 平台架构、Android 开发环境以及 Android 应用。

第 2 章　介绍了 Android 布局，主要包括线性布局、相对布局、表格布局、网格布局以及绝对布局等内容。

第 3 章　介绍了 Android 控件，主要包括文本类控件、按钮类控件、时钟类控件等内容。

第 4 章　介绍了 Android 视图，主要包括滚动视图、图片控件、列表视图、网格视图等内容。

第 5 章　介绍了 Android 动画，主要包括补间动画、帧动画、动画组件以及消息提示框等内容。

第 6 章　介绍了 Android 对话框，主要包括弹出式对话框、进度条对话框、日期时间选择对话框等内容。

第 7 章　介绍了 Android 菜单，主要包括选项菜单、单选复选菜单项、子菜单等内容。

第 8 章　介绍了 Android 程序组件，主要包括活动、服务、广播、消息、意图等内容。

第 9 章　介绍了 Android 通信，主要包括 TCP 通信、UDP 通信、HTTP 通信等内容。

第 10 章　介绍了 Android 开发，主要包括手机的附加功能、通话功能、短信功能、邮件功能以及定位功能等。

本书主要由左军编写，此外参加编写的还有刘超、邓俊辉、梁朗星、李旭波、张棣华、刘泳、邓耀隆、梁志成和周品。

由于作者的水平有限，加之时间紧凑，书中难免会存在不足之处，敬请广大读者批评指正。

编　者

2015 年 3 月

目　录

第 1 章　Android 初识

随着移动网络速度、移动设备性能的提升以及人们对移动设备功能要求的提高，Android 这一开放、快速、友好的手机操作系统应运而生已呈燎原之势。Android 是一种基于 Linux 的自由及开放源代码的操作系统，主要使用于移动设备，如智能手机和平板电脑，由 Google 公司于 2007 年 11 月 5 日发布的内核移动软件，该平台由操作系统、中间件、用户界面和应用软件组成，是一个真正开放的移动开发平台。

1.1　Android 简介

Android 为时尚美观、功能强大、尺寸多样的众多设备提供强力支持，这些设备运行的都是 Android 系统，因此可与用户的应用和 Google 产品完美配合。

1.1.1　Android 的发展史

Android 是一种以 Linux 为基础的开放源码操作系统，主要使用于便携设备。Android 主要发行了如下几个版本：

(1) Android 1.1 版本：在 2008 年 8 月发布的 Android 第一个版本。

(2) Android 1.5 版本：在 2009 年 4 月 30 日发布，命名为 Cupcake(纸杯蛋糕)。

版本主要更新有：

拍摄/播放影片，支持上传到 Youtube；支持立体声蓝牙耳机，同时改善自动配对性能；最新的采用 WebKit 技术的浏览器，支持复制、粘贴和页面搜索；GPS 性能大大提高；提供屏幕虚拟键盘；主屏幕增加音乐播放器和相框 widgets；应用程序自动随着手机旋转；短信、Gmail、日历，浏览器的用户接口大幅改进，如 Gmail 可以批量删除邮件，相机启动速度加快，拍摄图片可以直接上传到 Picasa；来电照片显示等。

(3) Android 1.6 版本：命名为 Donut(甜甜圈)，在 2009 年 9 月 15 日发布。

版本主要更新有：

重新设计的 Android Market 手势；支持 CDMA 网络；文字转语音系统(Text-to-Speech)；快速搜索框；全新的拍照接口；查看应用程序耗电；支持虚拟私人网络(VPN)；支持更多的屏幕分辨率；支持 OpenCore2 媒体引擎；新增面向视觉或听觉困难人群的易用性插件。

(4) Android 2.0 版本：在 2009 年 10 月 26 日发布。

版本主要更新有：

优化硬件速度；Car Home 程序；支持更多的屏幕分辨率；改良的用户界面；新的浏览器的用户接口和支持 HTML5；新的联系人名单；更好的白色/黑色背景比率；改进 Google

Maps3.1.2；支持 Microsoft Exchange；支持内置相机闪光灯；支持数码变焦；改进的虚拟键盘；支持蓝牙 2.1；支持动态桌面的设计。

(5) Android 2.2/2.2.1 版本：命名为 Froyo(冻酸奶)，在 2010 年 5 月 20 日发布。

版本主要更新有：

整体性能大幅度的提升；3G 网络共享功能；Flash 的支持；App2sd 功能；全新的软件商店；更多的 Web 应用 API 接口的开发。

(6) Android 2.3.x 版本：命名为 Gingerbread(姜饼)，在 2010 年 12 月 7 日发布。

版本主要更新有：

增加了新的垃圾回收和优化处理事件；原生代码可直接存取输入和感应器事件、EGL/OpenGLES、OpenSL ES；新的管理窗口和生命周期的框架；支持 VP8 和 WebM 视频格式，提供 AAC 和 AMR 宽频编码，提供了新的音频效果器；支持前置摄像头、SIP/VOIP 和 NFC(近场通讯)；简化界面、速度提升；更快更直观的文字输入；一键文字选择和复制/粘贴；改进的电源管理系统；新的应用管理方式。

(7) Android 3.0 版本：命名为 Honeycomb(蜂巢)，在 2011 年 2 月 2 日发布。

版本主要更新有：

优化针对平板；全新设计的 UI 增强网页浏览功能；n-app purchases 功能。

(8) Android 3.1 版本：命名为 Honeycomb(蜂巢)，在 2011 年 5 月 11 日发布。

版本主要更新有：

经过优化的 Gmail 电子邮箱；全面支持 Google Maps；将 Android 手机系统跟平板系统再次合并从而方便开发者；任务管理器可滚动，支持 USB 输入设备(键盘、鼠标等)；支持 Google TV.可以支持 XBOX 360 无线手柄；widget 支持的变化，能更加容易的定制屏幕 widget 插件。

(9) Android 3.2 版本：命名为 Honeycomb(蜂巢)，在 2011 年 7 月 13 日发布。

版本主要更新有：

支持 7 英寸设备；引入了应用显示缩放功能。

(10) Android 4.0 版本：命名为 Ice Cream Sandwich(冰激凌三明治)，在 2011 年 10 月 19 日在香港发布。

版本主要更新有：

全新的 UI；全新的 Chrome Lite 浏览器，有离线阅读，16 选项卡，隐身浏览模式等；截图功能；更强大的图片编辑功能；自带照片应用堪比 Instagram，可以加滤镜、加相框，进行 360 度全景拍摄，照片还能根据地点来排序；Gmail 加入手势、离线搜索功能，UI 更强大；新功能 People：以联系人照片为核心，界面偏重滑动而非点击，集成了 Twitter、Linkedin、Google+等通信工具。有望支持用户自定义添加第三方服务；新增流量管理工具，可具体查看每个应用产生的流量，限制使用流量，到达设置标准后自动断开网络。

(11) Android 4.1 版本：命名为 Jelly Bean(果冻豆)，在 2012 年 6 月 28 日发布。

版本主要更新有：

更快、更流畅、更灵敏；特效动画的帧速提高至 60fps，增加了三倍缓冲；增强通知栏；全新搜索；搜索将会带来全新的 UI、智能语音搜索和 Google Now 三项新功能；桌面插件自动调整大小；加强无障碍操作；语言和输入法扩展；新的输入类型和功能；新的连接类型。

(12) Android 4.2 版本：命名为 Jelly Bean(果冻豆)，在 2012 年 10 月 30 日发布。

Android 4.2 沿用“果冻豆”这一名称，以反映这种最新操作系统与 Android 4.1 的相似性，但 Android 4.2 推出了一些重大的新特性，具体如下：

Photo Sphere 全景拍照功能；键盘手势输入功能；改进锁屏功能，包括锁屏状态下支持桌面挂件和直接打开照相功能等；可扩展通知，允许用户直接打开应用；Gmail 邮件可缩放显示；Daydream 屏幕保护程序；用户连点三次可放大整个显示屏，还可用两根手指进行旋转和缩放显示，以及专为盲人用户设计的语音输出和手势模式导航功能等；支持 Miracast 无线显示共享功能；Google Now 现可允许用户使用 Gmail 作为新的数据来源，如改进后的航班追踪功能、酒店和餐厅预订功能以及音乐和电影推荐功能等。

(13) Android 4.4 版本：命名为 KitKat(奇巧巧克力)，2013 年 9 月 4 日凌晨，谷歌对外公布了 Android 新版本 Android 4.4 KitKat(奇巧巧克力)，并且于 2013 年 11 月 1 日正式发布，新的 4.4 系统更多地整合了自家服务，力求防止安卓系统继续碎片化、分散化。

1.1.2 Android 优势

Android 系统不断地优化自己的设计，并且一直保持着与其他手机操作系统相比而言的优点：开放性、平等性、无界性、方便性与硬件的丰富性。

1. 开放性

提到 Android 的优势，首先想到一定是其真正的开放性，其开放性包含底层的操作系统以及上层的应用程序等，Google 公司与开放手机联盟合作开发 Android 的目的就是建立标准化、开放式的移动单击软件平台，在移动产业内形成一个开放式的生成系统。

Android 的开放性也同样会使大量的程序开发人员投入到 Android 程序的开发中，这将为 Android 平台带来大量新的应用。

2. 平等性

在 Android 系统上，所有应用程序完全平等，系统默认自带的程序与自己开发的程序没有任何区别，程序开发人员可以开发个人喜爱的应用程序并替换掉系统程序，来构建个性化的 Android 手机系统，这些功能在其手机平台上是没有的。

在开发之初，Android 平台就被设计成由一系列应用程序组成的平台，所有的应用程序都运行在一个虚拟机上面。该虚拟机提供了系统应用程序之间和硬件资源通讯的 API。而除了该虚拟机，其他应用全部平等。

3. 无界性

Android 平台的无界性表现在应用程序之间的无界性，开发人员可以很轻松地将自己开发的程序与其他应用程序进行交互。例如，应用程序需要播放声音模块，而正好手机中已经有一个成熟的音乐播放器，此时就不需要再重复开发音乐播放功能，只需简单地加上几句话即可将成熟的音乐播放功能添加到自己的程序中。

4. 方便性

在 Android 平台中开发应用程序非常方便的，如果对 Android 平台比较熟悉，想开发一个功能全面的应用程序并不是什么难事。Android 平台为开发人员提供了大量的实用库及方便的工具，同时也将 Google Map 等强大的功能集成了进来，开发人员只需简单地调用几何代码可将强大的地图功能添加到自己的程序中。

5. 硬件的丰富性

由于平台的开放性，众多的硬件制造商推出各种各样、千奇百怪的产品，但这些产品功能

上的差异并不影响数据的同步与软件的兼容。例如，原来在诺基亚手机上的应用程序，可以很轻松地移到摩托罗拉手机上使用，并且联系人、短信等资料可以更方便地转移。

1.1.3 Android 平台架构

Android 系统是以 Linux 系统为基础的，Google 公司将其按照功能特性划分为 4 层，自下而上分别是 Linux 内核、中间件、应用程序框架和应用程序，如图 1-1 所示。

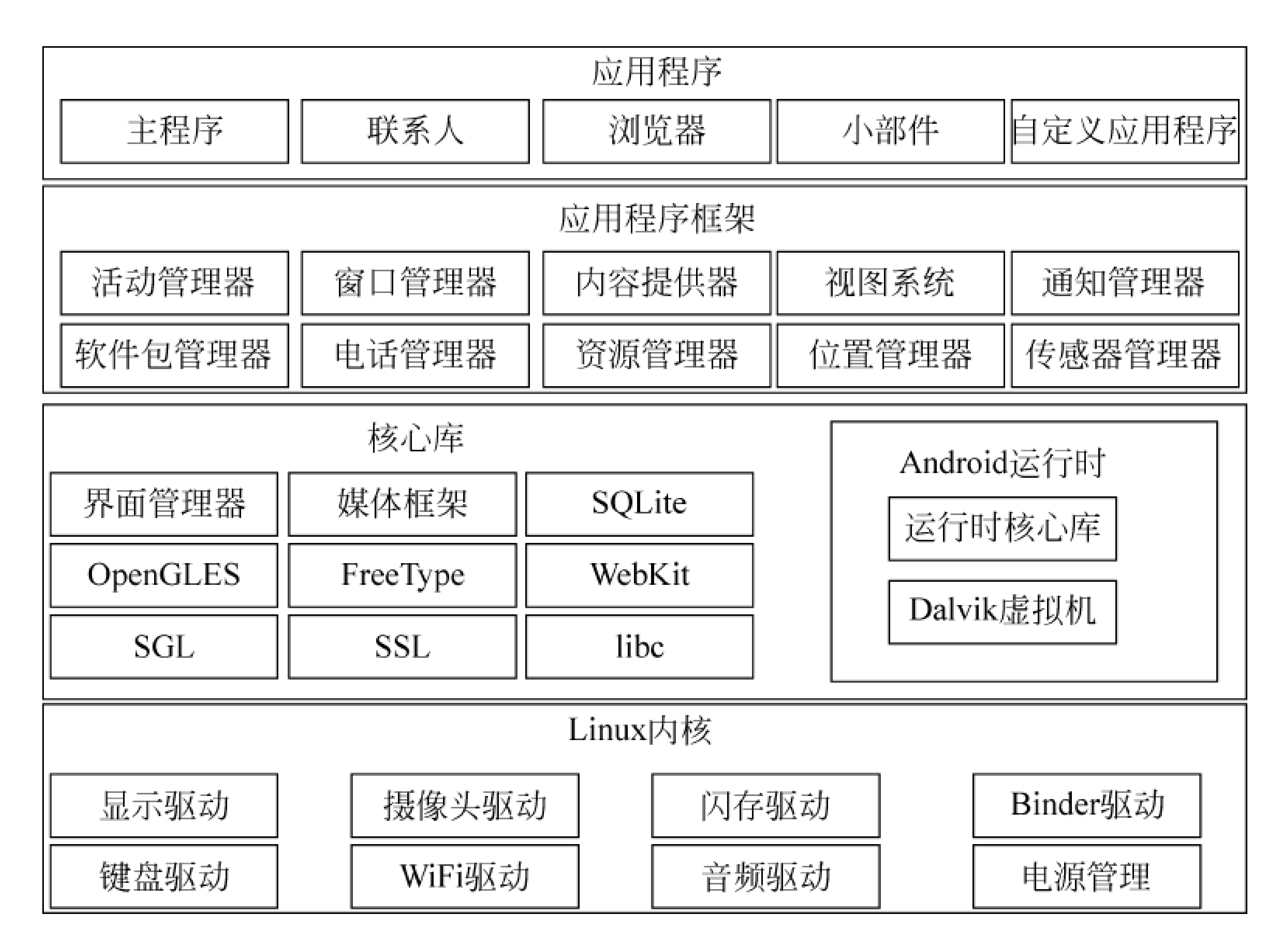


图 1-1　Android 系统框架图

1. 应用程序

Android 系统内置了一些常用的应用程序，包括 Home 视图、联系人、电话、浏览器等等。这些应用程序和用户自己编写的应用程序是完全并列的，同样都是采用 Java 语言编写的。用户可以根据需要增加自己的应用程序，或替换系统自带的应用程序。

2. 应用程序框架

应用程序框架提供了程序开发人员的接口，这是与 Android 程序员直接相关的部分，开发者可以用它开发应用。

- 丰富而又可扩展的视图（Views）：可以用来构建应用程序，包括列表（Lists），网格（Grids），文本框（Text Boxes），按钮（Buttons），甚至可嵌入的 Web 浏览器。
- 内容提供器（Content Providers）：使得应用程序可以访问另一个应用程序的数据（如联系人数据库），或共享它们自己的数据。
- 资源管理器（Resource Manager）：提供非代码资源的访问，如本地字符串、图形、布局文件（Layout Files）。
- 通知管理器（Notification Manager）：使得应用程序可以在状态栏中显示自定义的提示信息。
- 活动管理器（Activity Manager）：用来管理应用程序生命周期并提供常用的导航回退功能。

3. 中间件

中间件包括两部分：核心库(Libraries)和 Android 运行时环境(Android Runtime)。

1) 核心库

核心库中主要包括一些 C/C++核心库，方便开发者进行应用的开发。

- 系统 C 库(libc)：专门为基于 embedded linux 的设备定制的。
- 媒体库：支持多种常用的音频、视频格式回放和录制，同时支持静态图像文件。编码格式包括 MPEG4、H.264、MP3、AAC、AMR、JPG、PNG。
- SurfaceManager：对显示子系统的管理，并且为多个应用程序提供了 2D 和 3D 图层的无缝融合。
- WebKit/LibWebCore：Web 浏览引擎，支持 Android 浏览器和一个可嵌入的 Web 视图。
- SGL：底层的 2D 图形引擎。
- 3D libraries：基于 OpenGL ES 1.0 APIs 实现的 3D 引擎。
- FreeType：位图(Bitmap)和矢量(Vector)字体显示。
- SQLite：轻型关系型数据库引擎。

2) Android 运行时环境

Android 运行时环境主要包括 Android 核心库和 Dalvik 虚拟机。

- Android 核心库：提供了 Java 库的大多数功能。
- Dalvik 虚拟机：依赖于 Linux 内核的一些功能，如线程机制和底层内存管理机制。同时虚拟机是基于寄存器的，Dalvik 采用简练、高效的 byte code 格式运行，能够在低资耗和没有应用相互干扰的情况下并行执行多个应用。每一个 Android 应用程序都在它自己的进程中运行，都拥有一个独立的 Dalvik 虚拟机实例。Dalvik 虚拟机中可执行文件为.dex 文件，该格式文件针对小内存使用做了优化。所有的类都经由 Java 编译器编译，然后通过 SDK 中的 dx 工具转化成.dex 格式由虚拟机执行。

4. Linux 内核

Android 平台运行在 Linux 2.6 之上，其 Linux 内核部分相当于手机硬件层和软件层之间的一个抽象层。Android 的内核提供了显示驱动、摄像头驱动、闪存驱动、键盘驱动、WiFi 驱动、音频驱动和电源管理等多项功能。此外，Android 为了让 Android 程序可以用于商业目的，将 Linux 系统中受 GNU 协议约束的部分进行了取代。

1.1.4 Android 应用组成

Android 开发 4 大组件分别是：活动(Activity)，用于表现功能；服务(Service)，后台运行服务，不提供界面呈现；广播接收器(Broadcast Receiver)，用于接收广播；内容提供商(Content Provider)，支持在多个应用中存储和读取数据，相当于数据库。

1. 活动

Android 中，Activity 是所有程序的根本，所有程序的流程都运行在 Activity 之中，Activity 可以算是开发者遇到的最频繁，也是 Android 当中最基本的模块之一。在 Android 的程序当中，Activity 一般代表手机屏幕的一屏。如果把手机比作一个浏览器，那么 Activity 就相当于一个网页。在 Activity 当中可以添加一些 Button、Checkbox 等控件，可以看到 Activity 概念和网页的概念相当类似。

一般一个 Android 应用是由多个 Activity 组成的。这多个 Activity 之间可以进行相互跳转。例如，按下一个 Button 按钮后，可能会跳转到其他的 Activity。与网页跳转稍微有些不一样的是，Activity 之间的跳转有可能返回值。例如，从 Activity A 跳转到 Activity B，那么当 Activity B 运行结束的时候，有可能会给 Activity A 一个返回值。这样做在很多时候是相当方便的。

Android 四种的 Activity 加载模如图 1-2 所示。

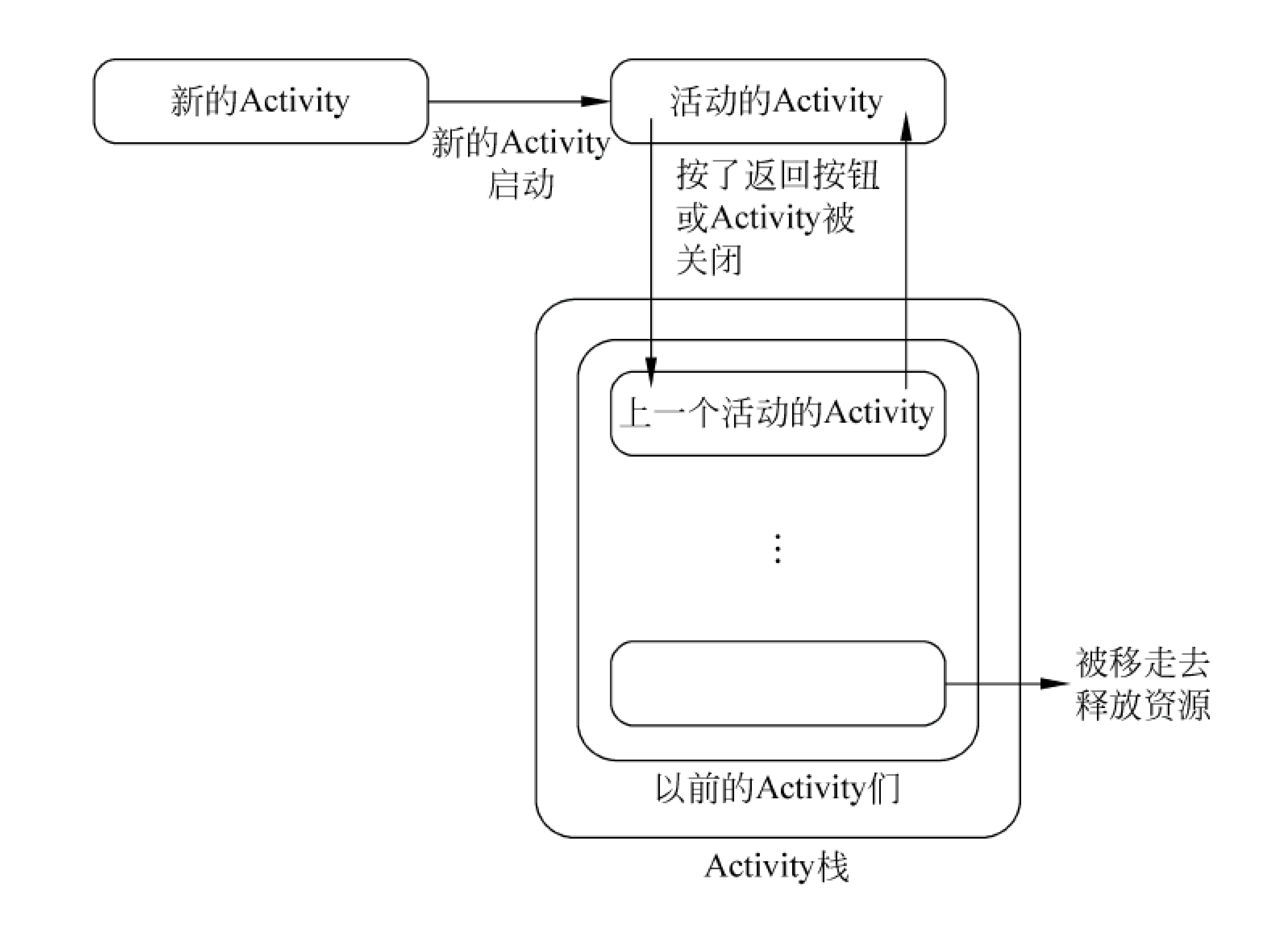


图 1-2　Android 四种 Activity 加载模流程图

当打开一个新的屏幕时，之前一个屏幕会被置为暂停状态，并且压入历史堆栈中。用户可以通过回退操作返回到以前打开过的屏幕。可以选择性的移除一些没有必要保留的屏幕，因为 Android 会把每个应用的开始到当前的每个屏幕保存在堆栈中。

2. 服务

Service 是 Android 系统中的一种组件，跟 Activity 的级别差不多，但是它不能自己运行，只能后台运行，并且可以和其他组件进行交互。Service 是没有界面长生命周期的代码。Service 是一种程序，可以运行很长时间，但是却没有用户界面。这么说有点枯燥，来看个例子。打开一个音乐播放器的程序，这时如果想上网，那么打开 Android 浏览器，这时虽然已经进入了浏览器这个程序，但是，歌曲播放并没有停止，而是在后台继续一首接着一首的播放。其实这个播放就是由播放音乐的 Service 进行控制。当然这个播放音乐的 Service 也可以停止。例如，当播放列表里边的歌曲都结束，或用户按下了停止音乐播放的快捷键等。Service 可以在很多场合的应用中使用，如播放多媒体时用户启动了其他 Activity，这时程序要在后台继续播放，比如检测 SD 卡上文件的变化，或在后台记录地理信息位置的改变等，而服务却藏在后台。

开启 Service 有两种方式：

(1) Context.startService()：Service 会经历 onCreate→onStart(如果 Service 还没有运行，则 Android 先调用 onCreate()，然后调用 onStart()；如果 Service 已经运行，则只调用 onStart()，所以一个 Service 的 onStart 方法可能会重复调用多次)；如果是调用者自己直接

退出而没有调用 StopService,服务会一直在后台运行。该服务的调用者再启动起来后可以通过 stopService 关闭服务。注意,多次调用 Context.startservice()不会嵌套(即使会有相应的 onStart()方法被调用),所以无论同一个服务被启动了多少次,一旦调用 Context.stopService()或 StopSelf(),都会被停止。

说明:传递给 StartService()的 Intent 对象会传递给 onStart()方法。调用顺序为 onCreate →onStart(可多次调用) →onDestroy。

(2) Context.bindService():服务会经历 onCreate()→onBind(),onBind 将返回给客户端一个 IBind 接口实例,IBind 允许客户端回调服务的方法,比如得到服务运行的状态或其他操作。这个时候把调用者(Context,如 Activity)会和服务绑定在一起,Context 退出了,服务就会调用 onUnbind→onDestroyed 相应退出,所谓绑定在一起即为"共存亡"了。

3. 广播接收器

在 Android 中,广播是一种广泛运用的在应用程序之间传输信息的机制。而广播接收器是对发送出来的广播进行过滤接受并响应的一类组件。可以使用广播接收器来让应用对一个外部的事件做出响应。例如,当电话呼入这个外部事件到来时,可以利用广播接收器进行处理。当下载一个程序成功完成时,仍然可以利用广播接收器进行处理。广播接收器不能生成 UI,也就是说对于用户来说不是透明的,用户是看不到的。广播接收器通过 NotificationManager 来通知用户这些事情发生了。广播接收器既可以在 AndroidManifest.xml 中注册,也可以在运行时的代码中使用 Context.registerReceiver()进行注册。只要是注册了,当事件来临时,即使程序没有启动,系统也在需要的时候启动程序。各种应用还可以通过使用 Context.sendBroadcast()将它们自己的 Intent 广播给其他应用程序。

4. 内容提供

内容提供(Content Provider)是 Android 提供的第三方应用数据的访问方案。

在 Android 中,对数据的保护是很严密的,除了放在 SD 卡中的数据,一个应用所持有的数据库、文件等内容,都是不允许其他直接访问的。Andorid 当然不会真的把每个应用都做成一座"孤岛",它为所有应用都准备了一扇窗,这就是 Content Provider。应用想对外提供的数据,可以通过派生 Content Provider 类,封装成一枚 Content Provider,每个 Content Provider 都用一个 uri 作为独立的标识,形如 content://com.xxxxx。所有应用看着像 REST 的样子,但实际上,它比 REST 更为灵活。和 REST 类似,uri 也可以有两种类型,一种是带 id 的;另一种是列表的,但实现者不需要按照这个模式来做,给 id 的 uri 也可以返回列表类型的数据。

1.2 Android 开发环境

在搭建环境前,需要了解安装开发工具所需要的硬件和软件配置条件。

1.2.1 Android 系统需求

本节介绍使用 Android SDK 进行开发所需的硬件和软件需求。对于硬件方面,要求 CPU 和内存尽量大。Android SDK 全部下载大概需要占用 4.5GB 硬盘空间。由于开发过程中需要反复重启模拟器,而每次重启都会消耗几分钟的时间(视机器配置而定),因此使用高配置的机器能节约不少时间。

支持 Android SDK 的操作系统及其要求如表 1-1 所示。

表 1-1 Android SDK 对操作系统的要求

操作系统	要　求
Windows	Windows XP(32 位)
	Vista(32 或 64 位)
	Windows7(32 位或 64 位)
Mac OS	10.5.8 或更新(仅支持 x86)
Linux(在 Ubuntu 的 10.04 版测试)	需要 GNU C Library(glibc)2.7 或更新
	在 Ubuntu 系统上,需要 8.04 版或更新
	64 位版本必须支持 32 位应用程序

对于开发环境,除了常用的 Eclipse IDE,还可以使用 IntelliJ IDEA 进行开发。对于 Eclipse 在下载 Android SDK 时就自带相兼容的版本。

1.2.2 安装 JDK

在 Windows 平台上,搭建 Android 开发环境,首先下载并安装与开发环境相关的软件资源,这些资源主要包括 JDK、Eclipse、Android SDK 和 Development Tools 插件(ADT 插件)。

1. 安装 JDK

在 Android 平台上,所有应用程序都是使用 Java 语言来编写的,所以要安装 Java 开发包 JDK(Java SE Development Kit),JDK 是 Java 开发时所必需的软件开发包。

安装 JDK 的过程比较简单,运行该程序后,根据安装提示选择安装路径,将 JDK 安装到指定的文件夹即可,默认安装目标为“C:\Program Files\Java\jdk1.6.0_10(jdk-6u10-rc2-bin-b32-windows-i586-p-12_sep_2008)”。

JDK 安装完毕后,要设置 Java 的环境变量,即设置 bin 和 lib 文件夹的路径。其操作步骤如下(操作系统为 Windows 7):

(1) 右击“计算机”,在弹出的快捷菜单中选择“属性”选项,在“系统”对话框中,单击“高级系统设置”按钮,弹出“系统属性”对话框,如图 1-3 所示。



图 1-3 “系统属性”对话框

（2）在“系统属性”对话框的“高级”选项卡中，单击“环境变量”按钮，弹出“环境变量”对话框，如图 1-4 所示。



图 1-4 “环境变量”对话框

（3）选中“系统变量”区域的 Path 变量，单击“编辑”按钮，弹出“编辑系统变量”对话框，如图 1-5 所示。

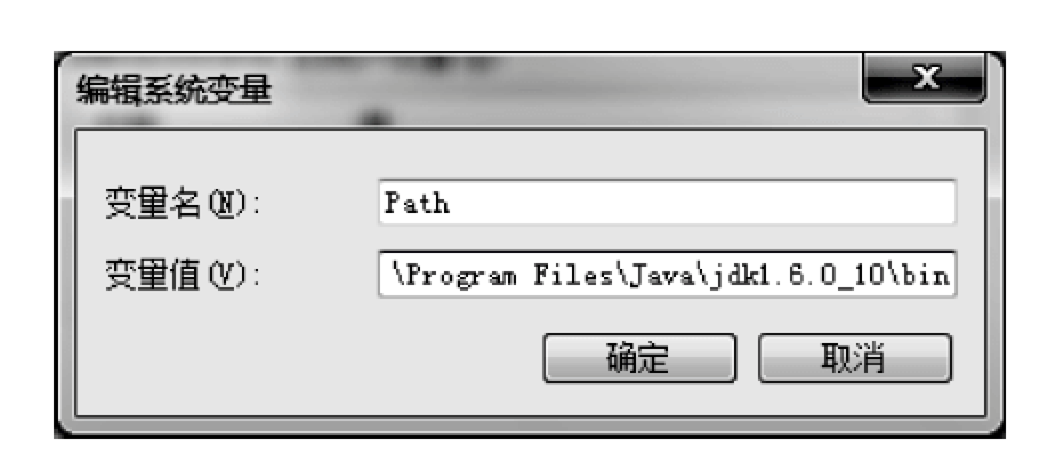


图 1-5 环境变量 Path 设置

（4）在该对话框的“变量值”文本框中添加“C:\Program Files\Java\jdk1.6.0_10\bin”，然后单击“确定”按钮即可完成设置，即设置了 bin 文件夹的路径。

（5）在“环境变量”对话框的“系统变量”区域，单击“新建”按钮，弹出“新建系统变量”对话框，如图 1-6 所示。



图 1-6 新建环境变量 classpath

（6）在图 1-6 中的“变量名(N)”文本框中输入 classpath，在“变量值(V)”文本框中输入“C:\Program Files\Java\jdk1.6.0_10\lib”，即可设置了 lib 文件夹的路径。

完成以上操作后，一个典型的 Java 开发环境便设置好了。在正式开始下一步前先验证

Java 开发环境的设置是否成功。

在 Windows 7 系统中单击“开始”按钮，在弹出的窗口中选择“运行”，在运行框中输入 cmd 并按 Enter 键，即可打开 CMD 窗口，在窗口中输入 java – version，则可显示所安装的 Java 版本信息，如图 1-7 所示。



图 1-7　JDK 安装成功页面

1.2.3　Eclipse 环境

从 Android 4.4 版本开始，下载的 Android 软件包中包括 Eclipse 软件，在官方网站下载相应 Android 软件包并解压即可看到 Eclipse 软件启动器，双击该软件，效果如图 1-8 所示。



图 1-8　Eclipse 启动界面

启动 Eclipse 开发环境桌面，将会看到选择工作空间的提示，如图 1-9 所示。

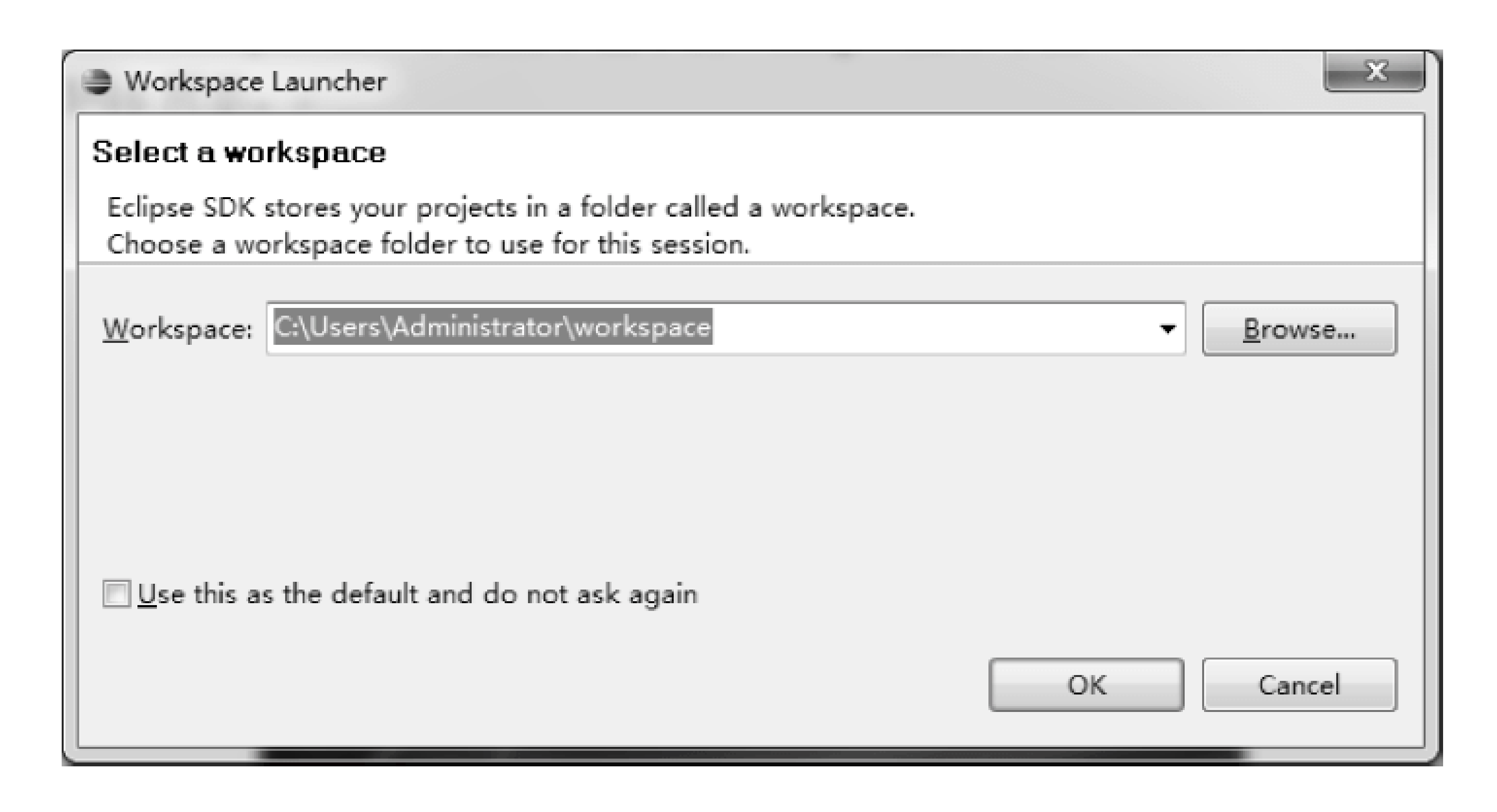


图 1-9　选择工作空间

接着单击图 1-9 中的 OK 按钮，即完成 Eclipse 的安装，系统进入 Eclipse 初始欢迎界面，如图 1-10 所示。接着单击左上角的“欢迎”按钮，即可进入 Eclipse 的开发环境界面，如图 1-11 所示。



图 1-10　Eclipse 欢迎界面

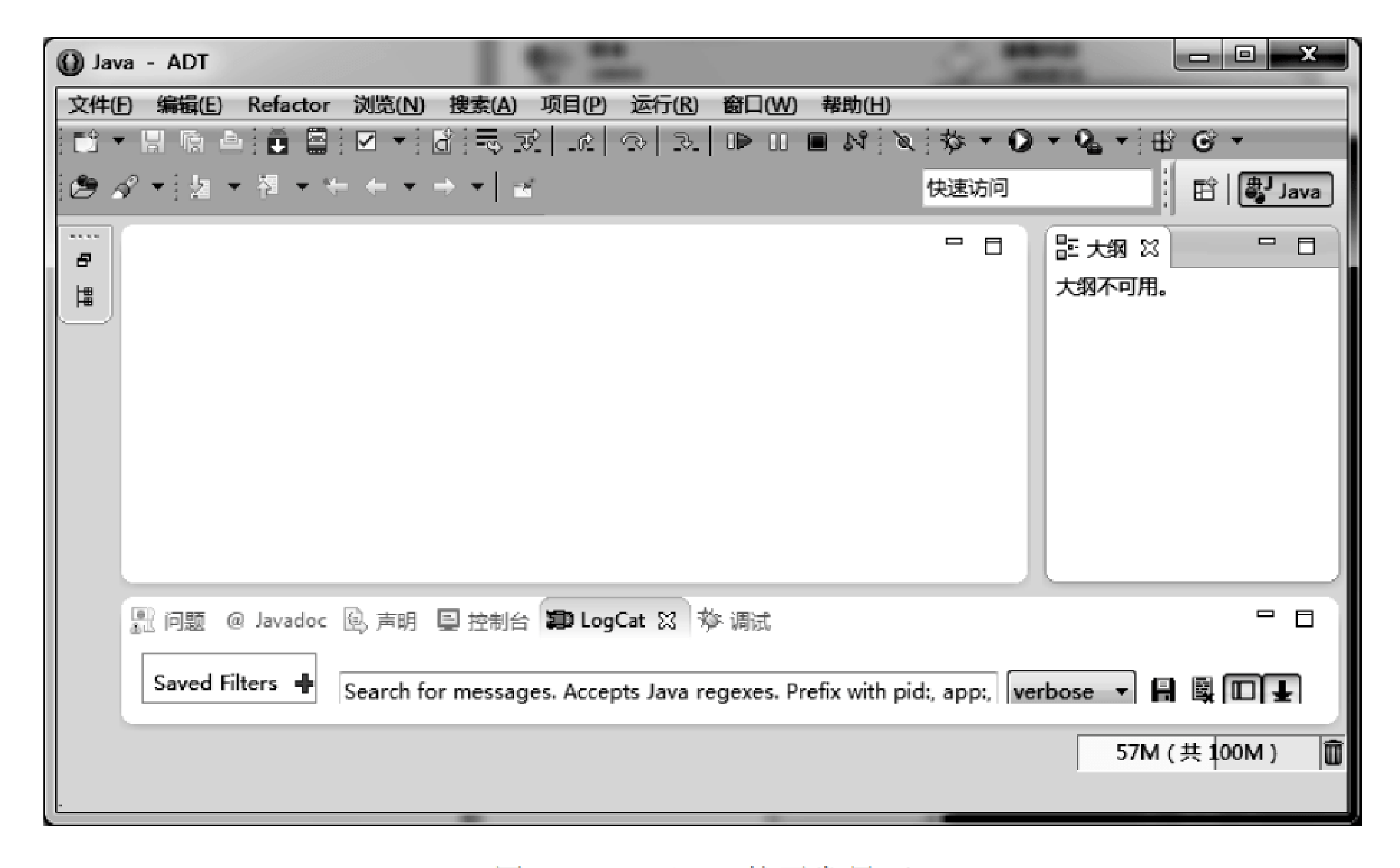


图 1-11　Eclipse 的开发界面

1.2.4　Android 的 ADK

（1）单击图 1-11 中的 快捷按钮，程序将自动检测是否有更新的 SDK 数据包可下载，如图 1-12 所示。

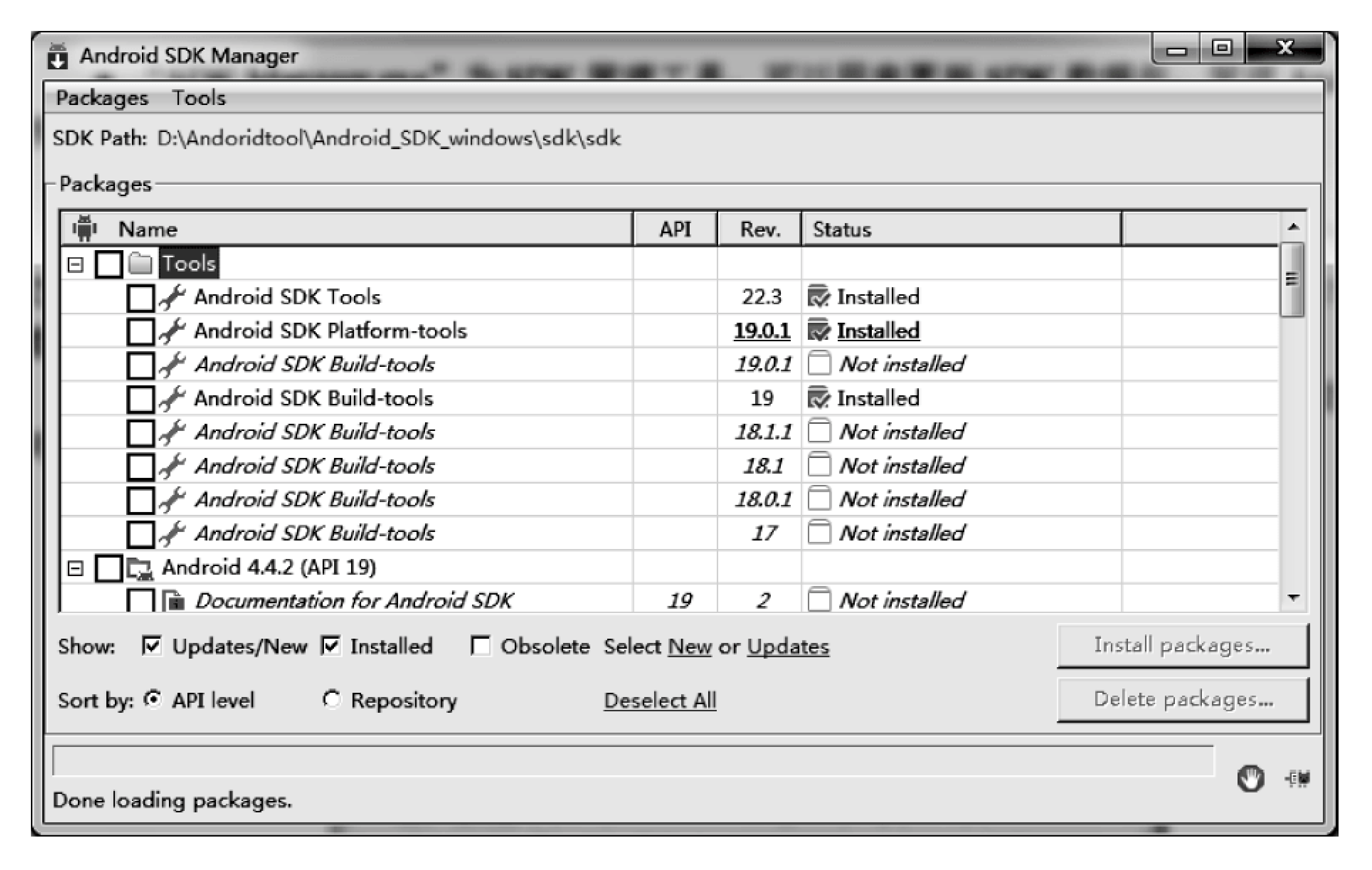


图 1-12　运行 SDK Manager.exe 执行文件

(2) 对于所要更新的内容,如果只想尝试一下 Android 4.4.2,那么选择 Android 4.4.2(API 19)项,然后单击 Install X packages 按钮来安装就可以了。如果要在此 SDK 上开发应用程序和游戏应用,那么需要接受并遵守所有许可内容(Accept All),并单击 Install 按钮。

(3) 接着将 SDK tools 目录的完整路径设置到系统变量 Path 中,这样便于在后面调用 Android 命令时,无须输入全部的绝对路径。设置系统变量 Path 的方法与 JDK 的环境变量值操作一致,在 Path 环境变量的"变量值(V)"文本框中添加";D:\Andoridtool\Android_SDK_windows\sdk\sdk\tools;"即可,如图 1-13 所示。



图 1-13　设置 Android SDK 环境变量

最后检查 Android SDK 是否安装成功并能够正常运行。在 Windows 7 系统中单击"开始"按钮,在弹出的窗口中选择"运行",在运行框中输入 cmd 并按 Enter 键,即可打开 CMD 窗口,在窗口中输入 android -h,则可显示所安装的 Android SDK 信息,如图 1-14 所示。

1.2.5　Android 的 AVD

AVD(Android Virtual Device)是 Android 运行的虚拟设备,是 Android 的模拟识别器。建立的 Android 要运行,必须创建 AVD,每个 AVD 上可以配置很多的运行项目。创建 AVD 时,可以配置的选项有模拟影像大小、触摸屏、轨迹球、摄像头、屏幕分辨率、键盘、GSM、GPS、Audio 录放、SD 卡支持、缓存大小等。

(1) 单击图 1-11 中的 快捷按钮,即可启动 Android AVD,弹出如图 1-15 所示的

```
管理员: C:\Windows\system32\cmd.exe
C:\Users\Administrator>android -h

       Usage:
       android [global options] action [action options]
       Global options:
  -h --help       : Help on a specific command.
  -v --verbose    : Verbose mode, shows errors, warnings and all messages.
     --clear-cache: Clear the SDK Manager repository manifest cache.
  -s --silent     : Silent mode, shows errors only.

                                                                Valid
                                                                actions
                                                                are
                                                                composed
                                                                of a verb
                                                                and an
                                                                optional
                                                                direct
                                                                object:
-    sdk              : Displays the SDK Manager window.
-    avd              : Displays the AVD Manager window.
-   list              : Lists existing targets or virtual devices.
-   list avd          : Lists existing Android Virtual Devices.
-   list target       : Lists existing targets.
-   list sdk          : Lists remote SDK repository.
```

图 1-14　Android SDK 安装成功信息

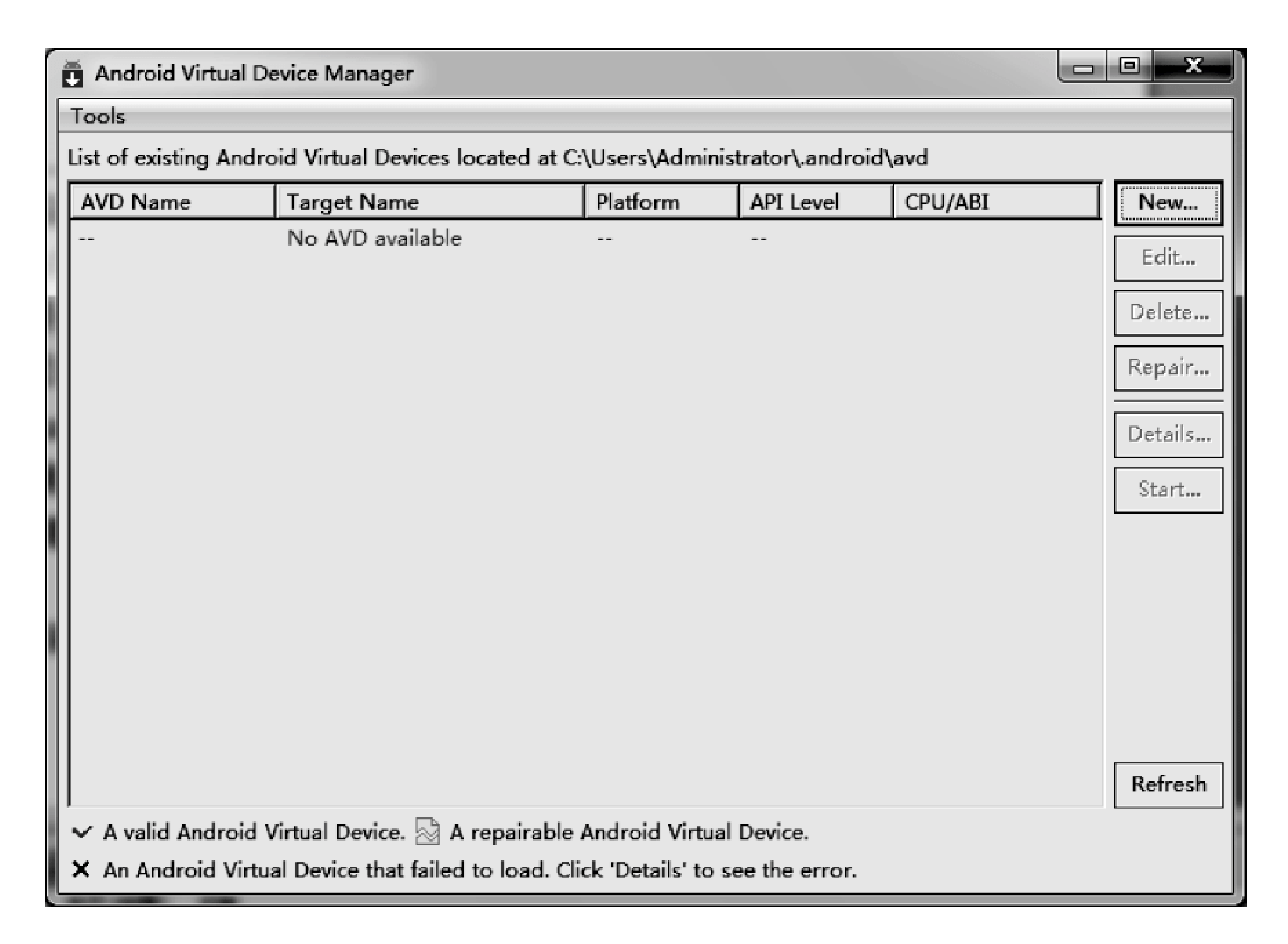


图 1-15　AVD Manager.exe 界面

Android Virtual Device Manager 窗口。

(2) 单击图 1-15 右侧的 New…按钮，弹出一个新的 Android Virtual Device (AVD)对话框，在该对话框中可以设置模拟器的配置，如图 1-16 所示。

- Name：创建 AVD 的名称。可以在文本框中输入所要创建的 AVD 的名称，注意名称中不能有空格符。
- Target：选择 Android 版本和 API 的等级。单击右边的下拉按钮，选择相应的 Android 版本和 API 的等级。

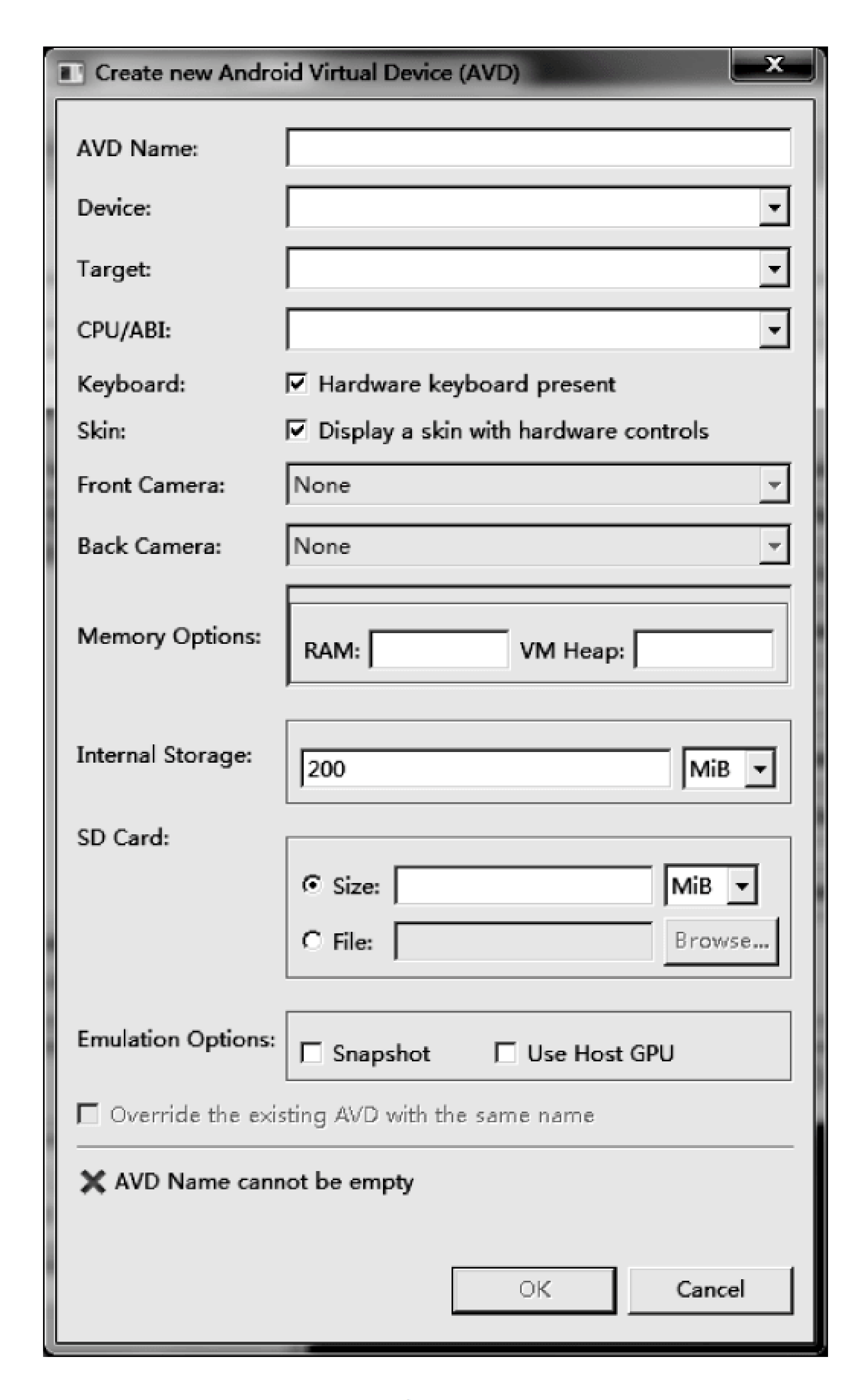


图 1-16　新建 AVD 时的设置

- SD Card：设置 SD 卡。在 Size 文本框中指定 SD 卡大小。另外，也可以在 File 文本框中设置已有的 SD 卡镜像文件的路径。
- Skin：设置模拟器的外观和屏幕分辨率。单击 Built-in 右边的下拉按钮，可以选择默认的 HVGA（320×480）、QVGA（240×320）、WVGA（480×800 或 480×854）、WQVGA（240×400 或 240×320）几种，在此选择默认的 HVGA（320×480）。另外，单击 Resolution 项，还可以自定义分辨率。不同版本的 Android 所设置的 Skin 参数有所不同。
- Hardware：设置模拟器支持的硬件设备的属性，包括影像大小、触摸屏、轨迹球、摄像头、屏幕分辨率、键盘、GSM、GPS、Audio 录放、SD 卡支持、缓存区大小等。单击该区域右边的 New…按钮，在弹出的对话框中可以设置各项的属性。

（3）设置好模拟器的参数后，单击图 1-16 下边的 OK 按钮即可创建一个 AVD。创建好的 AVD 将会显示在如图 1-17 所示的 Android Virtual Device Manager 对话框的文件列表中。

（4）选中所创建的 AVD 选项，单击右侧的 Start 按钮，弹出如图 1-18 所示的 Launch Options 对话框。

（5）单击 Launch 按钮即可成功启动 AVD，效果如图 1-19 所示。

使用同样的操作可以根据需要创建多个 AVD 模拟器。这样做的好处是，可以模拟程序在不同的 Android 版本上运行的兼容性。

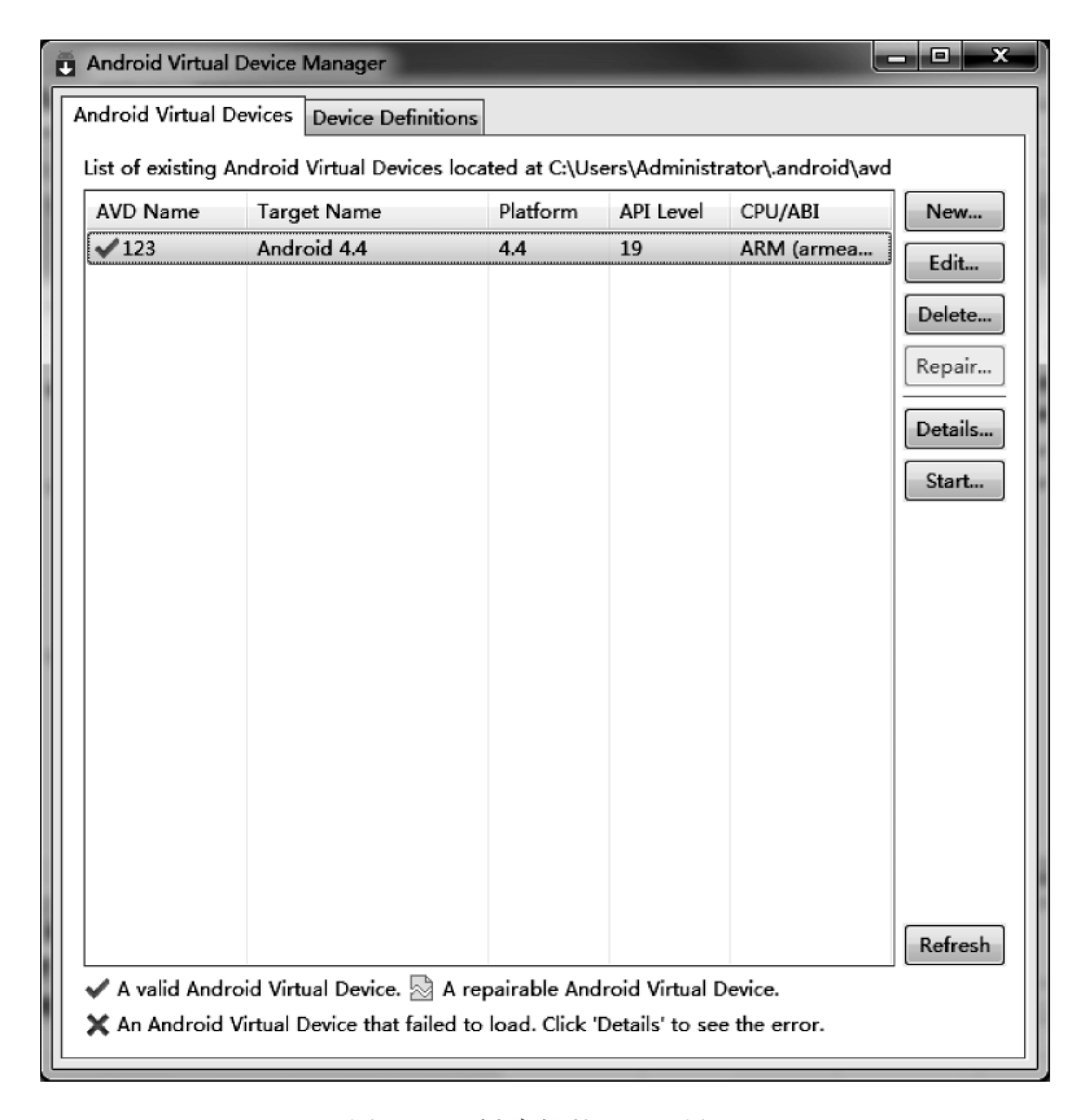


图 1-17　创建新的 AVD 界面

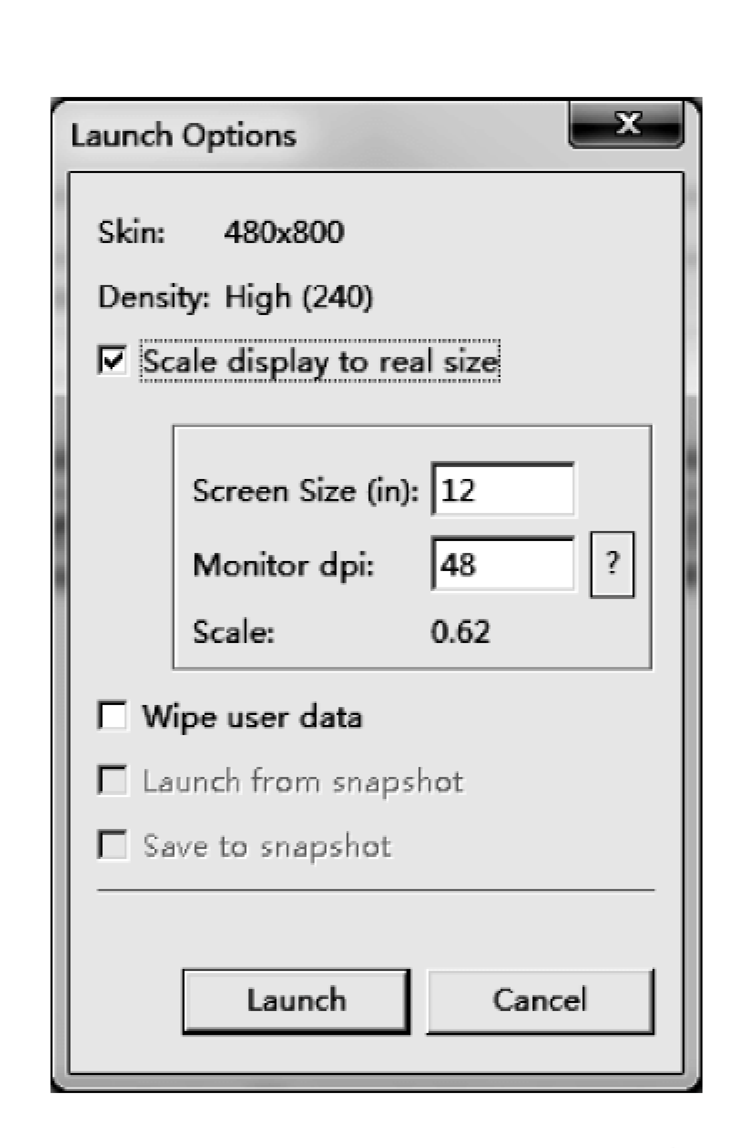


图 1-18　Launch Options 对话框



图 1-19　AVD 界面

图 1-19 右侧的各个控制按钮名称及其功能如表 1-2 所示。

表 1-2　AVD 的控制按钮功能

模拟器 AVD 的模拟按键	相应的图标	功　　能
音量渐小按钮		控制音量大小
电源按钮		设置电话模式，AVD 开关
问题是增加按钮		控制音量大小
上/下/左/右按钮		确定按钮
中心按钮		上/下/左/右移动焦点
Home 按钮		返回主界面
Menu 按钮		打开应用程序菜单
查询按钮		在手机内部或上网查询
返回按钮		返回上一级界面

1.3　Android 应用项目组成

Android 的应用项目主要由以下部分组成。

- src 目录：项目源文件都保存在这个目录中。
- R. java 文件：这个文件是 Eclipse 自动生成的，应用开发者不需要修改里面的内容。
- Android Library：应用运行的 Android 库。
- assets 目录：主要放置多媒体等一些文件。
- res 目录：主要放置应用会用到的资源文件。
- drawable 目录：主要放置应用会用到的图片资源。
- layout 目录：主要放置用到的布局文件。这结布局文件都是 XML 文件。
- value 目录：主要放置字符串(strings. xml)、颜色(colors. xml)、数组(arrays. xml)。
- AndroidManifest. xml：相当于应用的配置文件。在这个文件中，必须声明应用的名称，应用所用到的 Activity、Service 以及 receiver 等。

在 Eclipse 中，一个基本的 Android 项目的目录结构如图 1-20 所示。

1. src 目录

与一般的 Java 项目一样，src 目录下保存的是项目的所有包及源文件(. java)，res 目录下包含了项目中的所有资源。例如，程序图标(drawable)、布局文件(layout)和常量(value)等。不同的是，在 Java 项目中没有 gen 目录，也没有每个 Android 项目特有的 AndroidManifest. xml 文件。

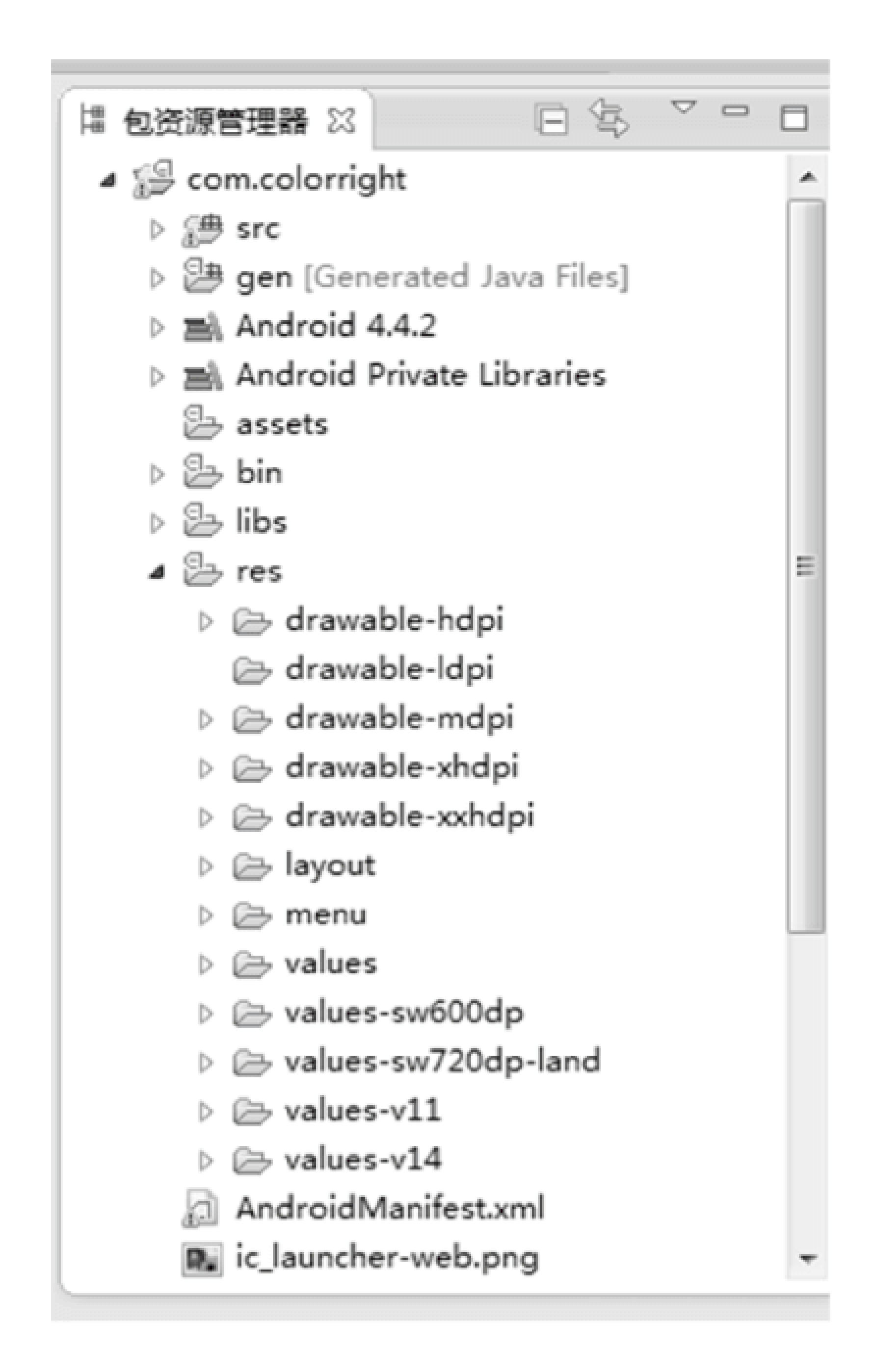


图 1-20　Android 应用工程文件组成

java 格式文件是在建立项目时自动生成的，这个文件是只读模式，R.java 文件是定义该项目所有资源的索引文件，其代码如下：

```
package fs.helloworld;
public final class R {
    public static final class attr {
    }
    public static final class dimen {
        public static final int activity_horizontal_margin = 0x7f040000;
        public static final int activity_vertical_margin = 0x7f040001;
    }
    public static final class drawable {
        public static final int ic_launcher = 0x7f020000;
    }
    public static final class id {
        public static final int action_settings = 0x7f080000;
    }
    public static final class layout {
        public static final int main = 0x7f030000;
    }
    public static final class menu {
        public static final int main = 0x7f070000;
    }
    public static final class string {
        public static final int action_settings = 0x7f050001;
        public static final int app_name = 0x7f050000;
        public static final int hello_world = 0x7f050002;
    }
    public static final class style {
        public static final int AppTheme = 0x7f060001;
    }
}
```

从上述代码中，可以看到文件定义了很多常量，并且会发现这些常量的名字都与 res 文件夹中的文件名相同，这再次证明.java 文件中所存储的是该项目所有资源的索引。有了这个文件，在程序中使用资源将变得更加方便，可以很快地找到要使用的资源，由于这个文件不能手动编辑，所以当用户在项目中加入了新的资源时，只需要刷新该项目，.java 文件便自动生成了所有资源的索引。

2. res 目录

在 res 目录下包含了该项目所使用到的资源文件，这里面的每一个文件或资源都将在 R.java 文件中进行索引定义。主要包括如下几类文件。

- 图片文件：分别提供了高分辨率(drawable-hdpi)、低分辨率(drawable-ldpi)、中分辨率(drawable-mdpi)、超高分辨率(drawable-xhdpi)、超高清分辨率(drawable-xxhdpi)的图片文件。
- 布局文件：在 layout 目录下，默认只有一个 main.xml，用户也可以添加更多的布局文件。
- 字符串：在 values 目录下的 strings.xml 文件中。

打开 main.xml 布局文件，其代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
```

```
        android:layout_height = "match_parent"
        android:paddingBottom = "@dimen/activity_vertical_margin"
        android:paddingLeft = "@dimen/activity_horizontal_margin"
        android:paddingRight = "@dimen/activity_horizontal_margin"
        android:paddingTop = "@dimen/activity_vertical_margin"
        tools:context = ".MainActivity" >
        <TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "@string/hello_world" />
</RelativeLayout>
```

在该布局文件中，首先定义了相对布局，内部只有一个文本框控件。这个控件显示了内容引用了 string 文件中 hello 变量。

其中：

- <RelativeLayout></RelativeLayout>：相对版面配置，在这个标签中，所有元件都是按相对排队排成的。
- android:layout_width：定义当前视图在屏幕上所占的宽度，fill_parent 即填充整个屏幕。
- android:layout_height：随着文字栏位的不同而改变这个视图的宽度或高度。
- android:paddingBottom：指屏幕界面底部的填充方式。
- android:paddingLeft：指屏幕界面左侧的填充方式。
- android:paddingRight：指屏幕界面右侧的填充方式。
- android:paddingTop：指屏幕界面顶部的填充方式。
- tools:context：该布局文件所调用的 Activity 内容。

在上述布局代码中，使用了一个 TextView 来配置文件标签 Widget(构件)，其中设置的属性 android:layout_width 为整个屏幕的宽度，android:layout_height 可以根据文字来改变高度，而 android:text 则设置了这个 TextView 要显示的文字内容，这里引用了@string 中的 hello 字符串，即 String. xml 文件中的 hello 所代表的字符串资源。hello 字符串的内容“HelloWorld、HelloAndroid”这就是用户在 HelloAndroid 项目运行时看到的字符串。

Strings. xml 文件的代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <string name = "app_name"> Hello World </string>
    <string name = "action_settings"> Settings </string>
    <string name = "hello_world"> Hello world! </string>
</resources>
```

3. AndroidManfest. xml 文件

在文件 AndroidManfest. xml 中包含了该项目中所使用的 Activity、Service、Receiver，以下代码为 HelloWorld 项目中的 AndroidManfest. xml 文件。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"      //根节点
    package = "fs.helloworld"                                               //包名
    android:versionCode = "1"
    android:versionName = "1.0" >
    <uses - sdk
```

```
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />                //SDK 版本
    < application                                         //图标和应用程序名称
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        < activity
            android:name = "fs.helloworld.MainActivity"  //默认启动的 Activity
            android:label = "@string/app_name" >         //Activity 名称
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" />
                < category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
        </activity >
    </application >
</manifest >
```

1.4 Android 应用

在 Eclipse 中,可以非常便捷地创建、调试 Android 应用程序。下面创建一个简单的 Android 应用项目。

1.4.1 创建 Android 应用项目

创建一个 Android 应用项目,其实现步骤如下。

1. 新建工作空间

一般创建的应用项目都需要存盘,或是 C 盘空间,所创建的文件需要放置到其他盘上,其实现步骤为:

(1) 在 Eclipse 主界面中选择"文件/切换工作空间(W)/其他"选项,效果如图 1-21 所示。

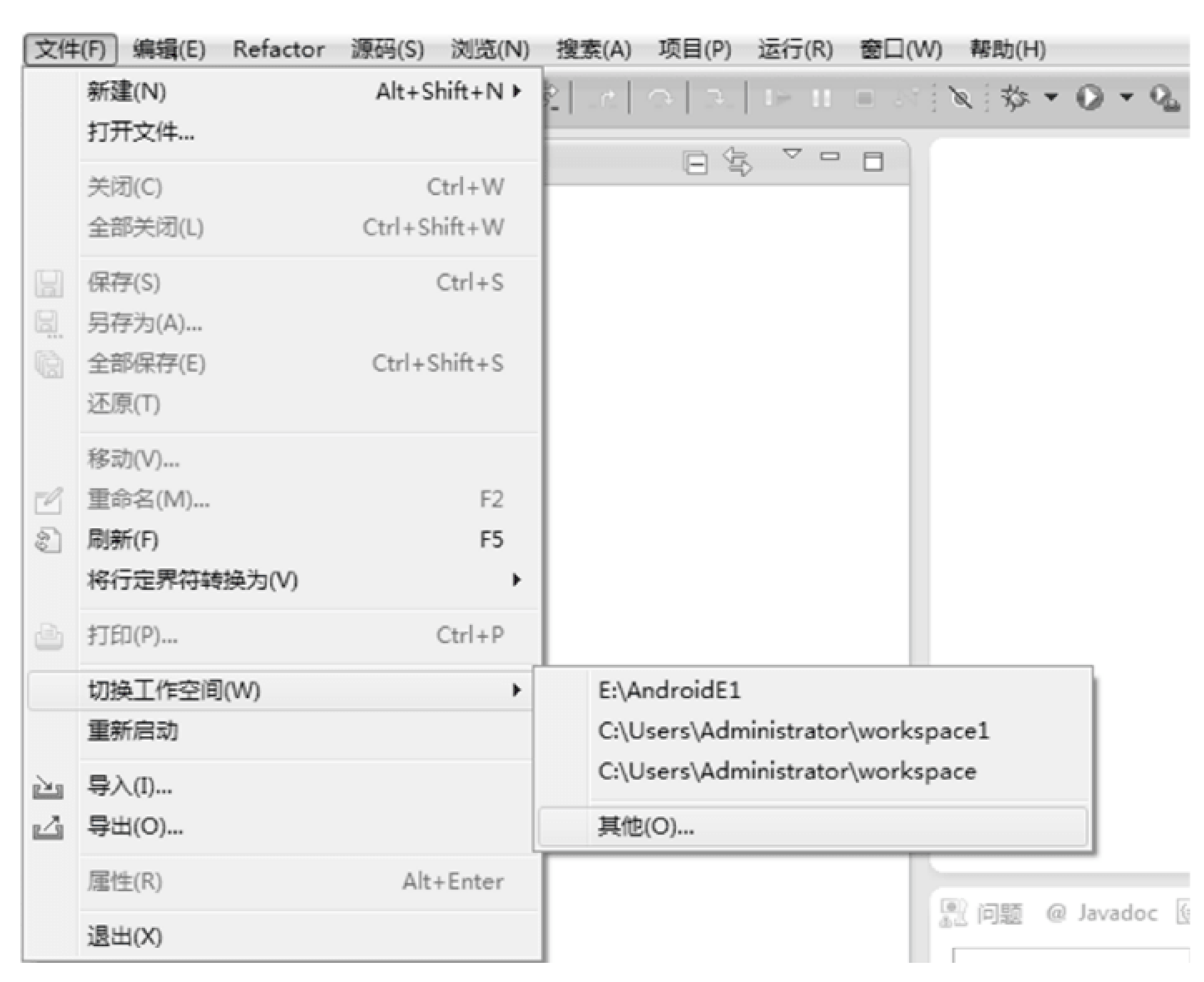


图 1-21 切换工作空间操作

（2）弹出“工作空间启动程序”对话框，在“工作空间（W）”右侧的文本框中输入所需要的路径，如把文件放在E盘新建的example2文件夹中，如图1-22所示。

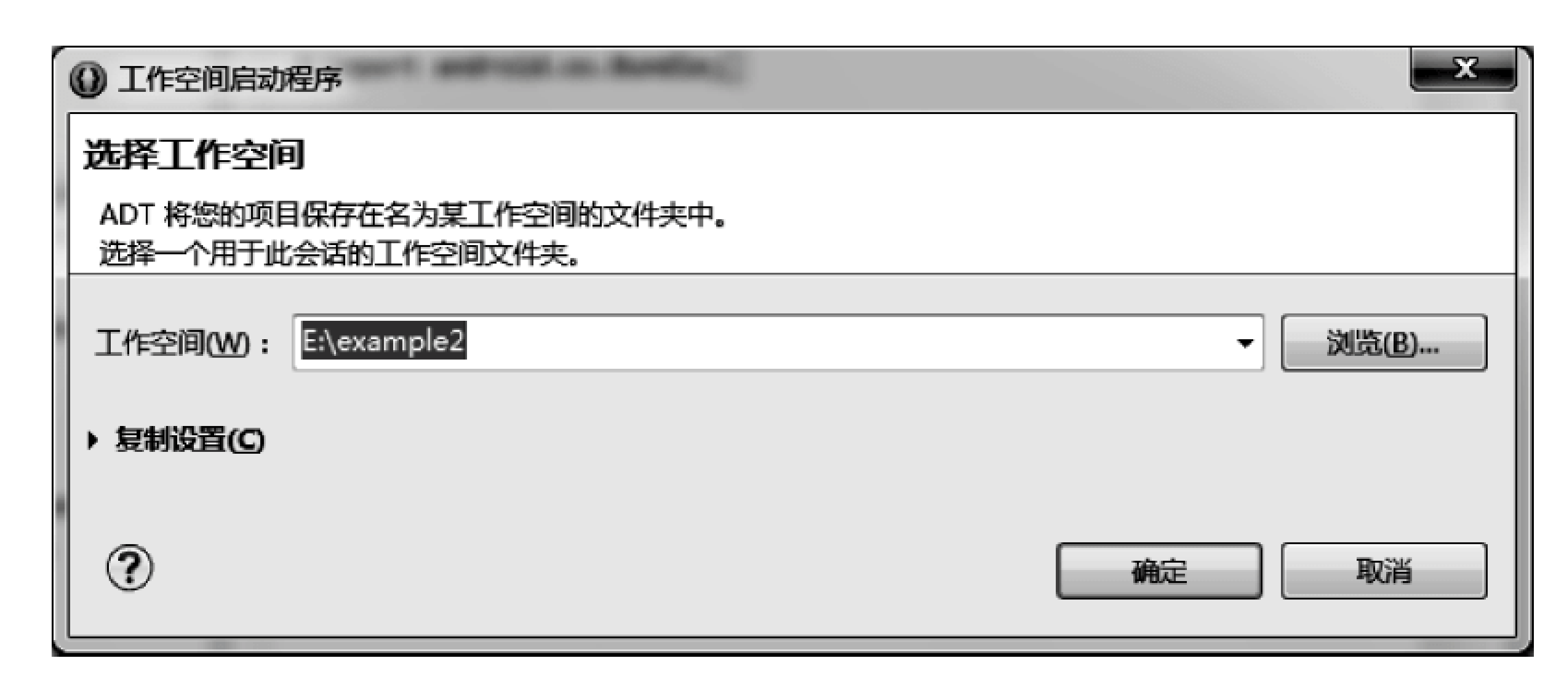


图1-22 “工作空间启动程序”对话框

（3）设置存盘路径后，单击图1-22中的“确定”按钮即可完成工作空间的创建，也即是说以后所创建的项目文件都在该文件夹中。

（4）完成工作空间启动程序设置后，Eclipse会自动重新启动。

2. 创建Android项目

（1）在Eclipse主界面中，单击“新建”图标可创建一个Android应用项目，弹出如图1-23所示的窗口。



图1-23 新建对话框

（2）选择图1-23中的Android Application Project项，单击“下一步”按钮，弹出如图1-24所示的创建Android项目页面。

应用项目命名填写规则为：

Application Name：填写应用程序的名称。默认情况下，会将前面填写的项目名称填写在此处，可以进行修改。该名称将作为应用程序的名称出现在手机应用列表中。

Project Name：该栏为工程项目的名称，即在Eclipse工作空间创建的文件夹名称，一般

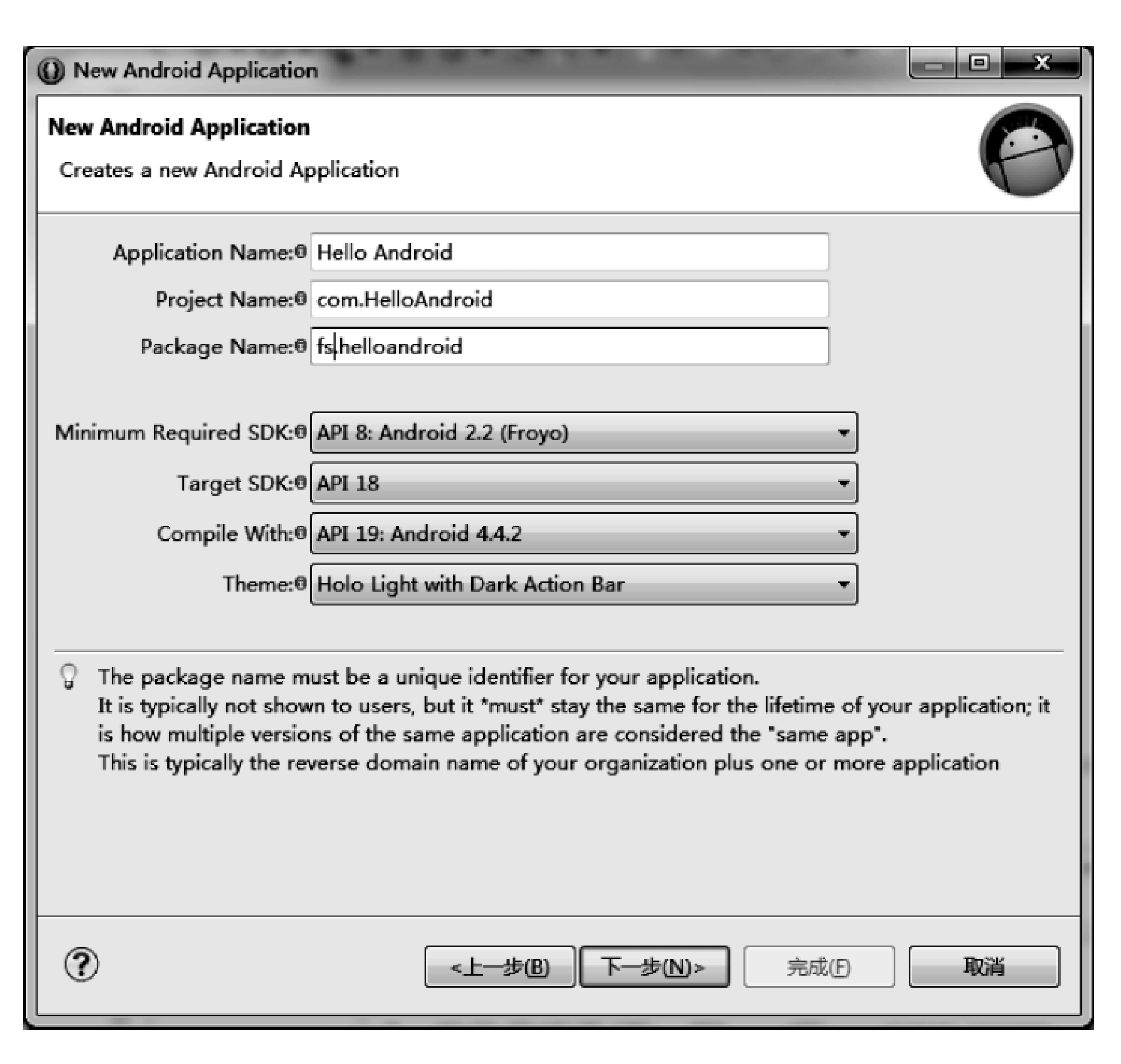


图 1-24　创建 Android 项目页面

以 com. * 形式命名。

Package Name：Java 源文件的包名，Eclipse 会自动在 src 下创建该包名。该包名一般是以 * . * 形式命名。

(3) 单击图 1-24 中的“下一步”按钮，弹出如图 1-25 所示的页面，在该页面中可以确定所创建的 Android 应用项目的内容，如自定义图标、Activity、项目保存的工作空间等。



图 1-25　“确定创建应用程序”页面

（4）单击“下一步”按钮，弹出如图 1-26 所示的页面，在该页面中可以自定义应用的图标。

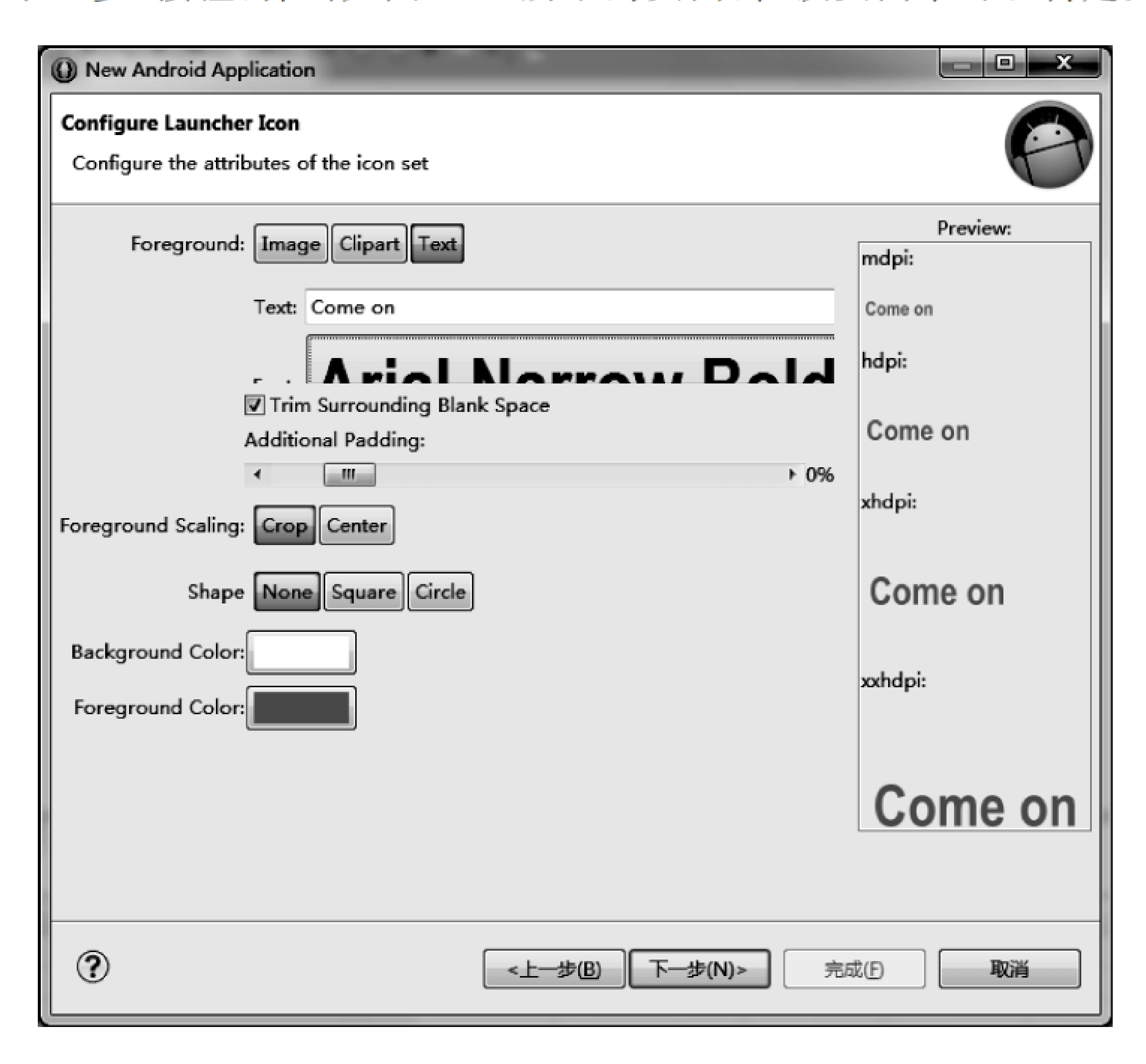


图 1-26　自定义图标页面

（5）单击“下一步”按钮，弹出如图 1-27 所示的页面，在该页面中创建一个空白的 Android 程序。

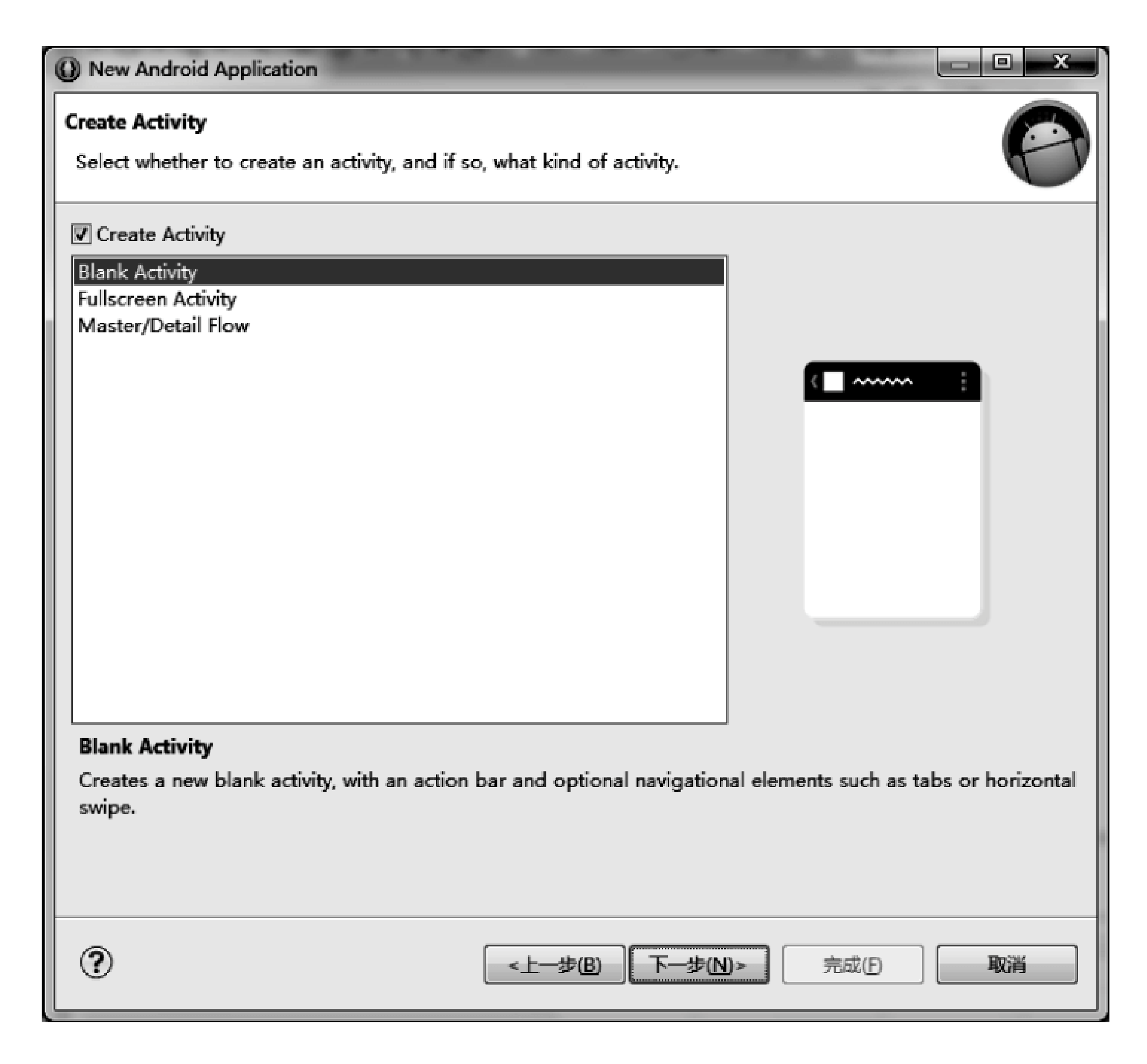


图 1-27　空白的 Android 程序

（6）单击“下一步”按钮，弹出如图 1-28 所示的页面，在该页面中可重新命名主活动名称、布局文件名称以及导航的类型。

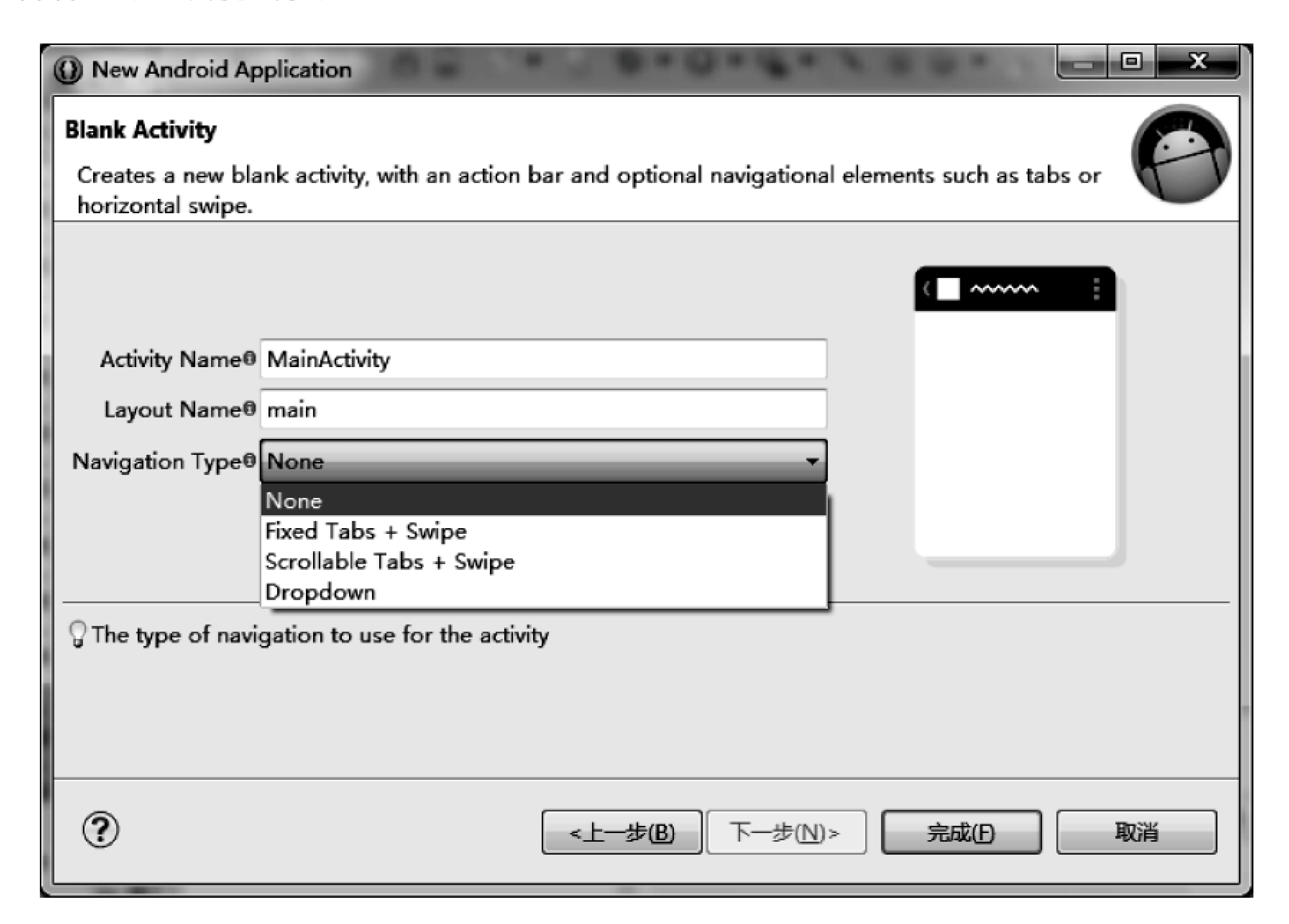


图 1-28　重命名界面

（7）单击“下一步”按钮，即可完成 Android 应用项目的创建，效果如图 1-29 所示。

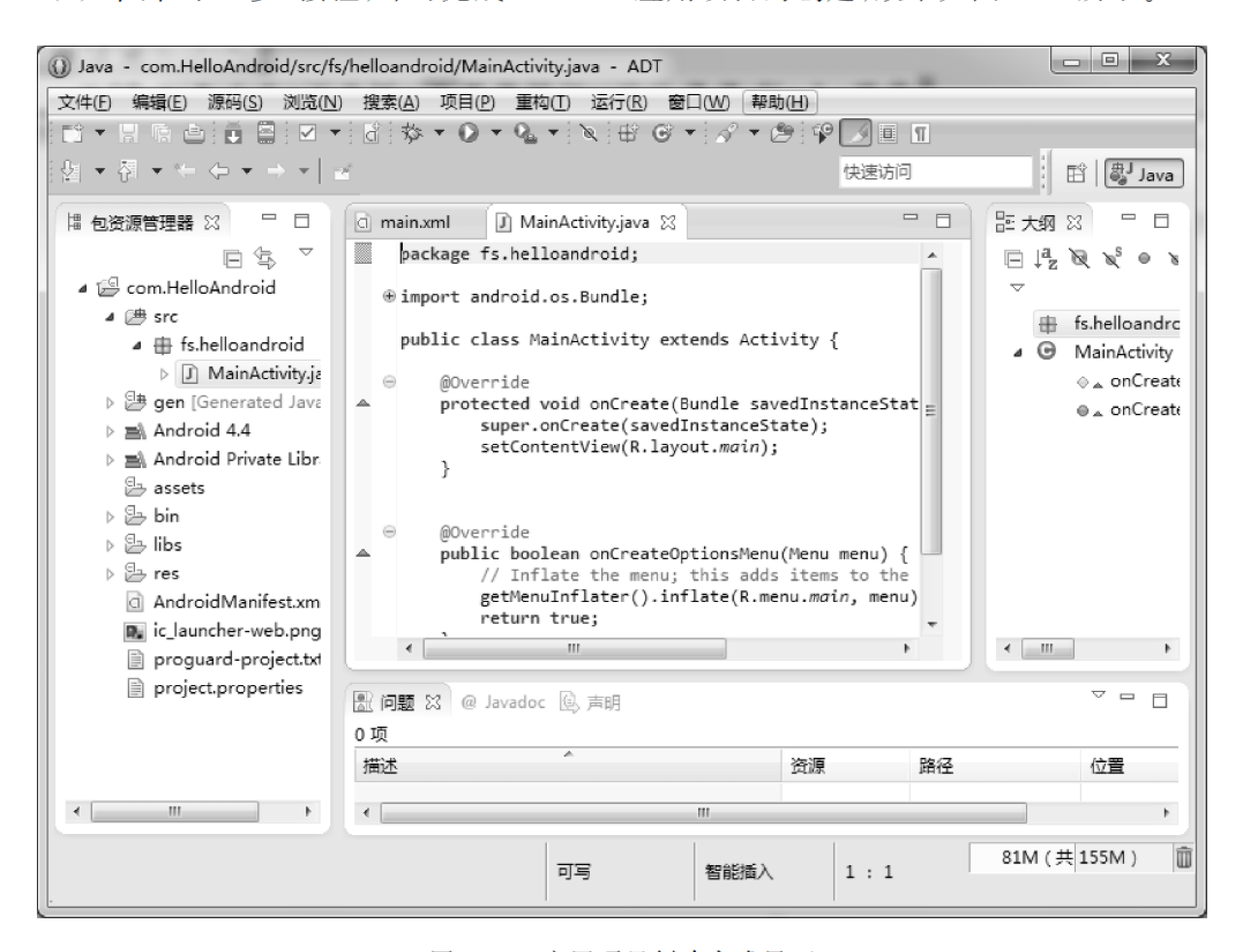


图 1-29　应用项目创建完成界面

图 1-29 中左侧的树形窗口为 HelloAndroid 应用项目的根目录，中间为应用项目的主活动程序，右侧为应用项目的大纲提要。

(8) 默认创建的应用项目的 res\layout 目录下的 main.xml 布局文件的，默认代码为：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
</RelativeLayout>
```

在文件中定义一个相对布局 RelativeLayout，并在布局文件中声明一个 TextView 控件。

应用项目的 src\fs.helloandroid 包下的 MainActivity.java 主活动文件，默认代码为：

```
package fs.helloandroid;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

(9) 单击图 1-29 中的"运行"按钮图标，或选中程序并右击，在弹出的快捷菜单中选择"运行方式(R)/Android Application"选项，即可弹出如图 1-30 所示的运行方式。



图 1-30　选择运行方式

(10) 在图 1-30 中选择 Android Application 选项，并单击下方的"确定"按钮，即可运行 com.HelloAndroid 项目，效果如图 1-31 所示。

1.4.2　DDMS 使用

在 Android SDK 工具中，提供了 DDMS(Dalvik Debug Monitor Service)来对 Android 的应用程序进行调试和模拟服务，主要提供了针对特定的进程查看正在运行的线程以及堆信息、输入日志(Logcat)、广播状态信息、模拟电话呼

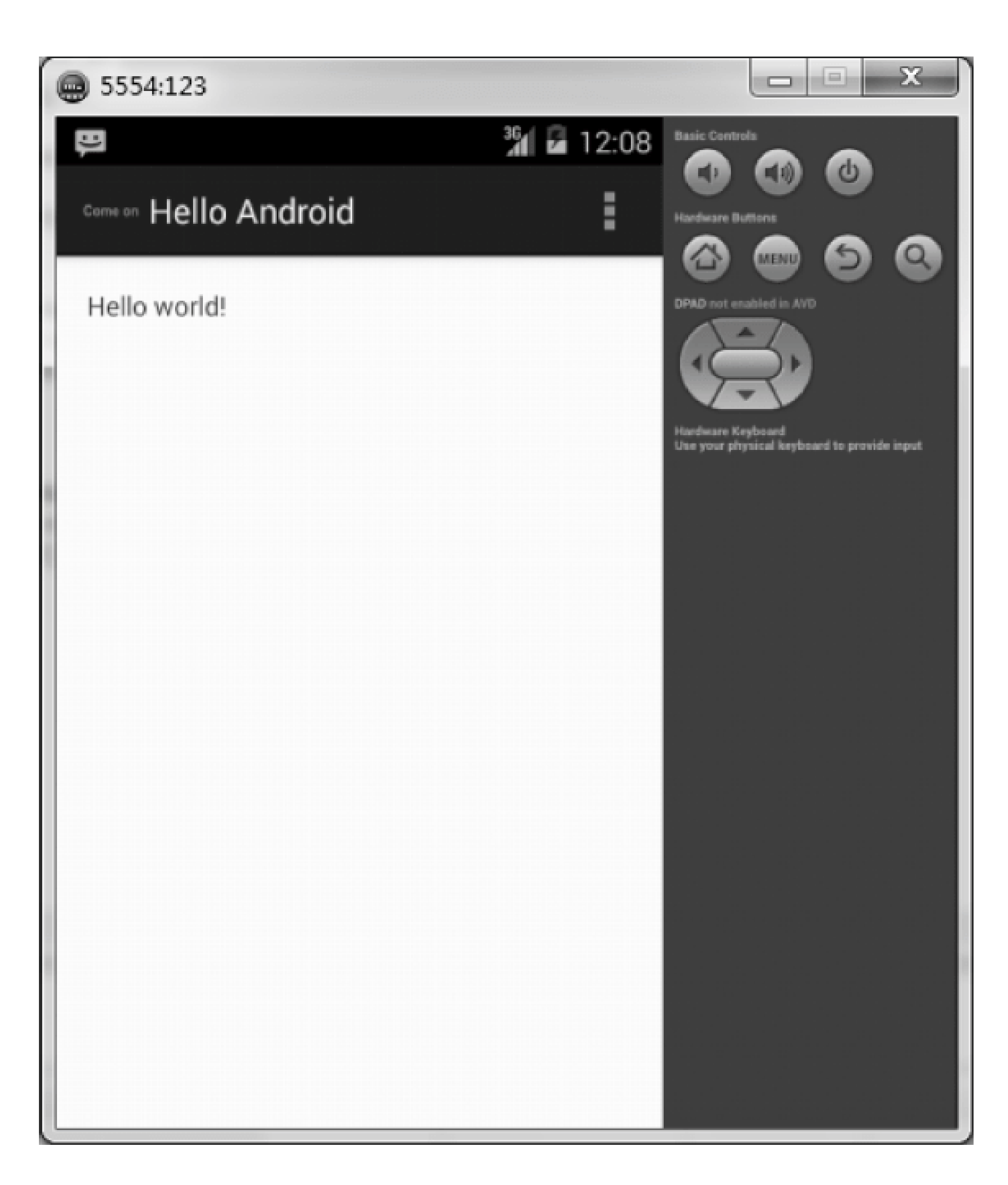


图 1-31　显示界面

叫、接收 SMS、虚拟地理坐标、为测试设备截屏等。

DDMS 会搭建 Eclipse 本地与测试终端(模拟器或真实设备)的链接,它们应用各自建立的端口监听调试器的信息,DDMS 可以实时监测到测试终端的连接情况。当有新的测试终端连接后,DDMS 将捕捉到终端的 ID,并通过 adb 工具建立调试器,从而实现发送指令到测度终端的目的。

1. 开启 DDMS 视图

在 Eclipse 的右上角有个 DDMS 图标,单击该图标即可打开 Eclipse 中所有的视图界面。除此之外还可以在 Eclipse 的菜单栏中选择“窗口/打开透视图”选项,在弹出的菜单中选择“其他”选项,效果如图 1-32 所示,也即可打开所有视图。选择图 1-32 中的 DDMS,切换到 DDMS 界面。



图 1-32　打开透视图

2. DDMS 功能

在 DDMS 视图界面中,有调试 Android 设备经常使用的工具,主要包括设备(Devices)、模拟器控制台(Emulator Control)、日志输出(LogCat)、文件目录(File Explorer)以及线程、堆栈等。这些功能都显示在 DDMS 界面中。如果在 DDMS 界面中没有找到这些功能选项,则在 Eclipse 界面菜单栏中选择“窗口/显示视图”选项,选择“其他”选项,将会出现 Eclipse 中所有的功能视图,如图 1-33 所示,选择需要的功能视图进行说明。

在 DDMS 提供的功能中，最常用的主要有以下 4 个。

(1) 设备(Devices)：设置功能视图一般在 DDMS 的左上角，其标签为 Devices，如图 1-34 所示。在该视图中显示所有连接的 Android 设备并且详细列出该 Android 设备中可连接调试的应用程序进程。从该图可看出列表中从左到右分别为应用程序名以及调试器链接的端口号。在进行调试时，一般只需要关心应用程序名。

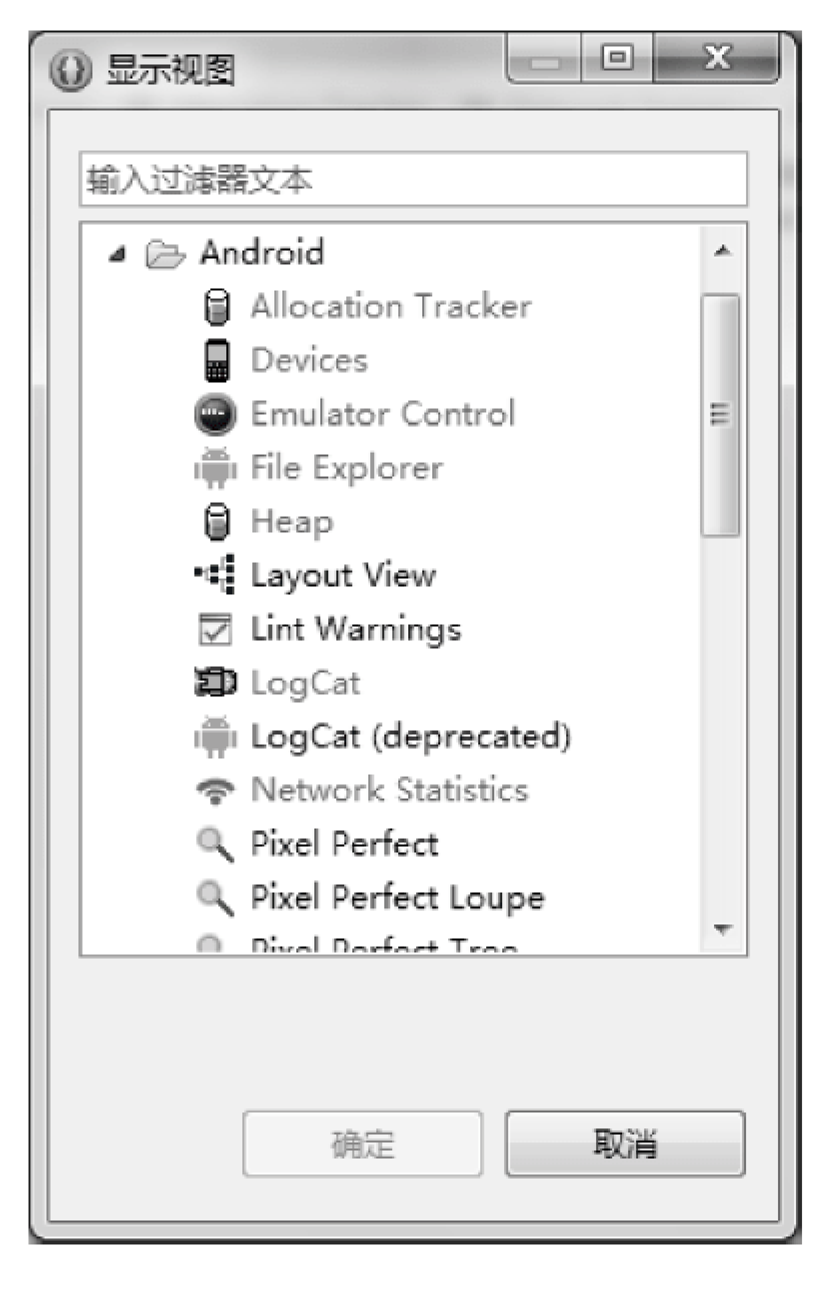


图 1-33　功能视图

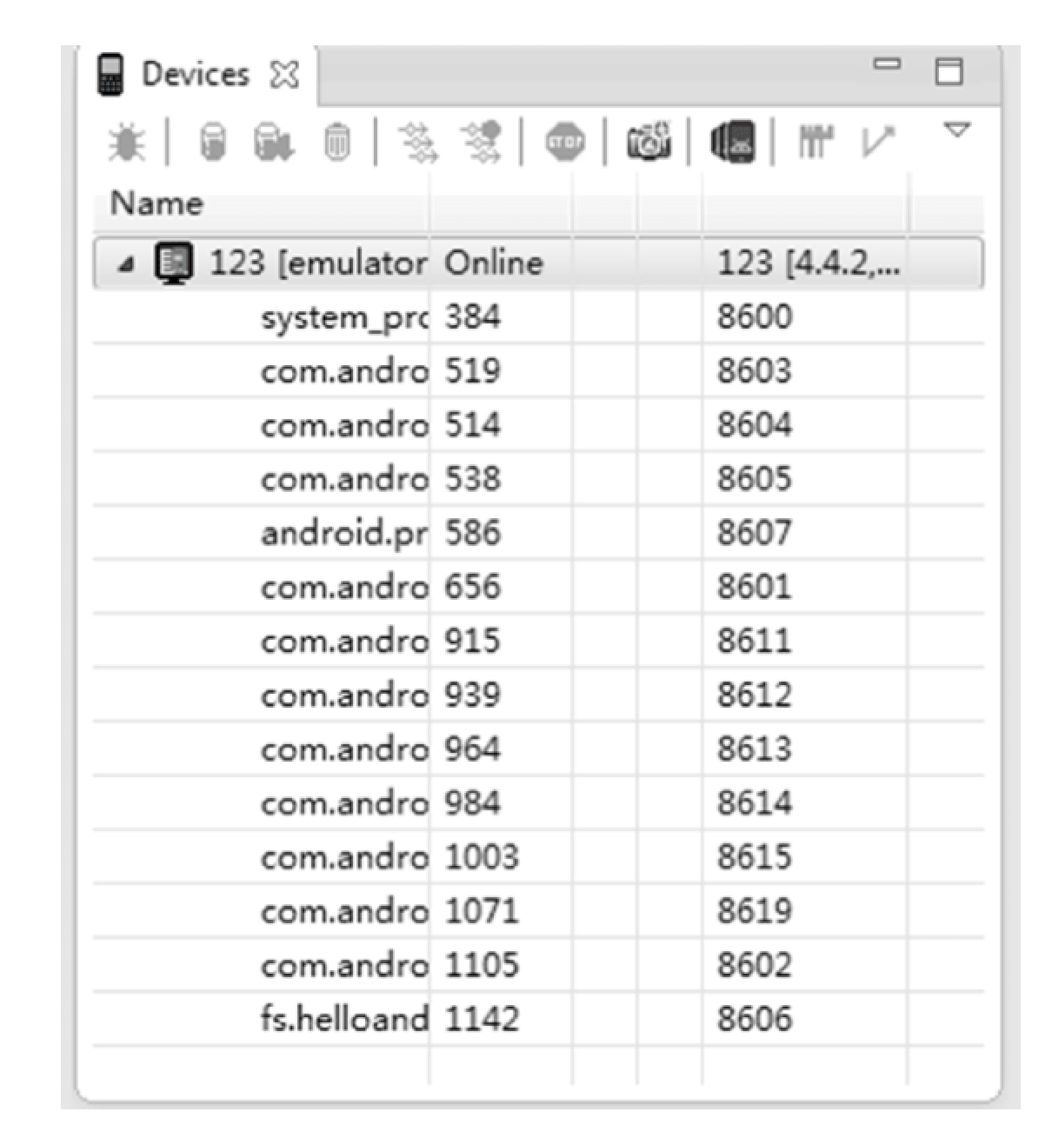


图 1-34　设备列表

当选择了列表中的某一个应用程序时，在视图的右上角有一排功能按钮即可使用。它们主要用于调试某个应用，主要的功能有调试选项(Debug the Selected Process)、线程查看(Update Threads)、堆栈查看(Update Heap)、终止进程(Stop Process)和截屏(Screen Shot)。

- Debug the Selected Process：用于显示被选择进程与调试器连接状态。如果进程前带有绿色表示该进程的源文件在 Eclipse 中处于打开状态，并已经开启了调试器监听进程运行情况。
- Update Threads：用于查看当前进程所包含的线程。当选中任意进程后，单击该按钮后，被选中的进程名称后边会出现显示线程信息标识并可以在 Threads 功能界面中看到详细的线程运行情况。
- Update Heap：用于查看当前进程堆栈内存的使用情况。当选中任意进程后，单击该按钮，可在 Heap 功能界面中看上去详细的堆栈使用情况，与 Update Threads 类似。
- Stop Process：终止当前进程。选择进程后，单击该按钮便强制终止了该进程。
- Screen Shot：截取当前测度终端桌面。

(2) 模拟器控制台(Emulator Control)：由于在模拟器中不断直接使用电话、短信、GPS 位置等功能，当使用模拟区测试这些功能时，可以通过该控制台来实现对这些交互功能的模拟。模拟器控制台视图如图 1-35 所示。

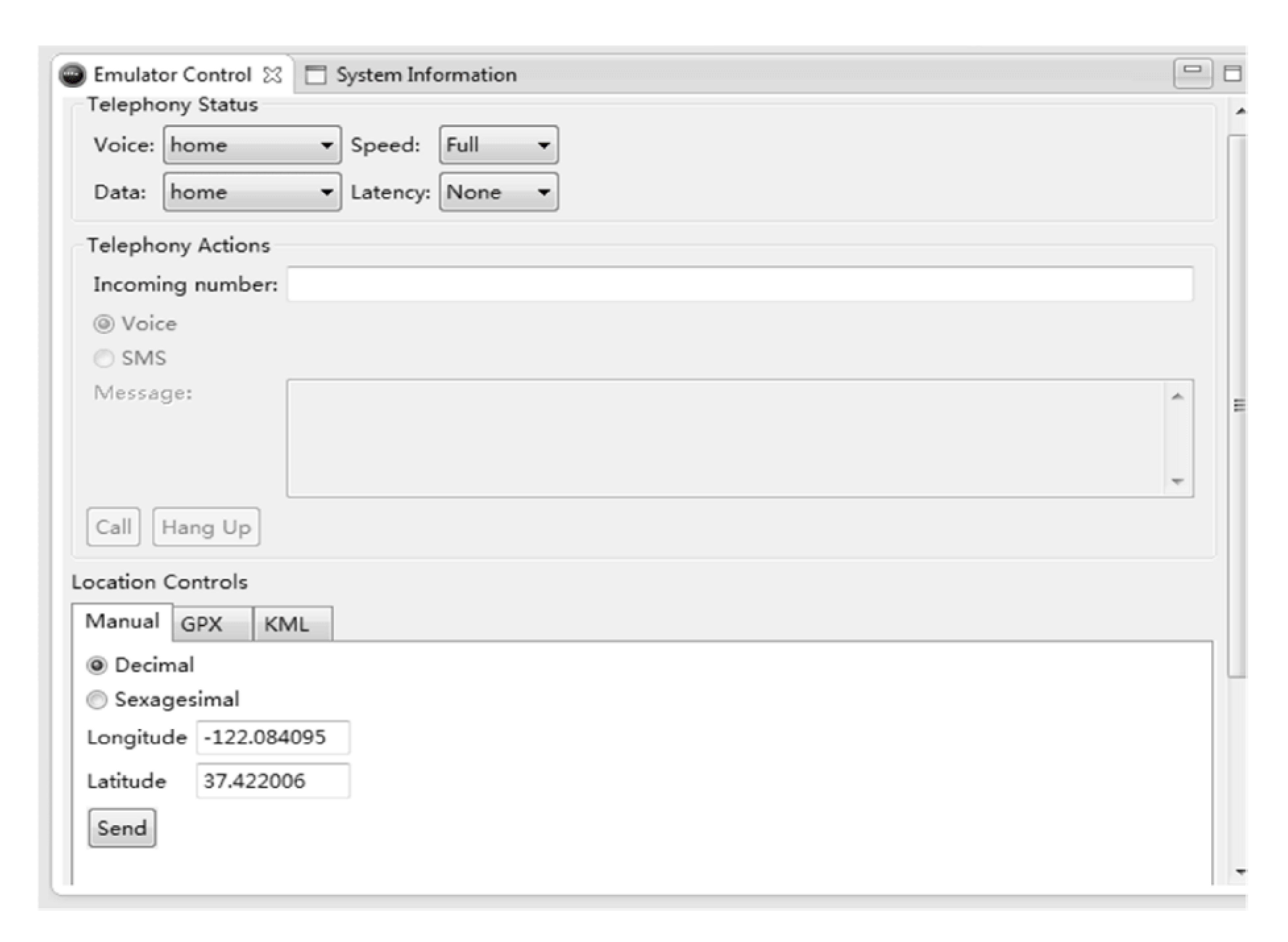


图 1-35 控制台

其各选项说明为：

- Relephony Status：选择模拟语音质量以及信号连接模式。
- Telephony Actions：模拟电话呼入和发送短信到测试的模拟器。其中 Incoming number 为设置本地呼叫模拟器的号码；Voice 选项表示模拟电话呼入模拟器；SMS 选项表示模拟短信发送到模拟器中。
- Location Controls：模拟地理坐标或模拟动态的路线坐标变化并显示预设的地理标识。其中，有三个选项卡表示可以使用不同的三种方式，即 Manually 方式，手动为终端发送二维经纬坐标；GPX 方式，通过 GPX 文件导入序列动态变化地理坐标，从而模拟行 GPS 变化的数值；KML 方式，通过 KML 文件导入独特的地理标识，并以动态形式根据变化的地理坐标显示在测度终端。

(3) 文件目录(File Explorer)：在 DDMS 界面的右边，占用较大一块区域的便是模拟器运行的详细信息，有多个选项卡，其中 File Explorer 为文件目录，如图 1-36 所示。

在文件目录中显示 Android 设备的文件系统信息。一般情况下，File Explorer 会有 data、mnt 和 system 三个目录。

- data 目录对应手机的 RAM，会存放 Android 系统运行时的 Cache 等临时数据。如果没有 roots 权限，apk 程序将会安装在/data/app 中(只是存放 apk 文件本身)；在/data/data 中存放着所有程序(系统应用程序和第三方应用程序)的详细数据目录信息。
- 在 mnt 目录中最重要的是其目录下的 sdcard 目录。该目录即对应于 SD Card 的目录文件。
- 在 system 目录中对应手机的 ROM，存放 Android 系统以及系统自带的应用程序等。

除了可以查看到这三个目录外，还可以使用 File Explorer 来对文件进行操作。选项卡右

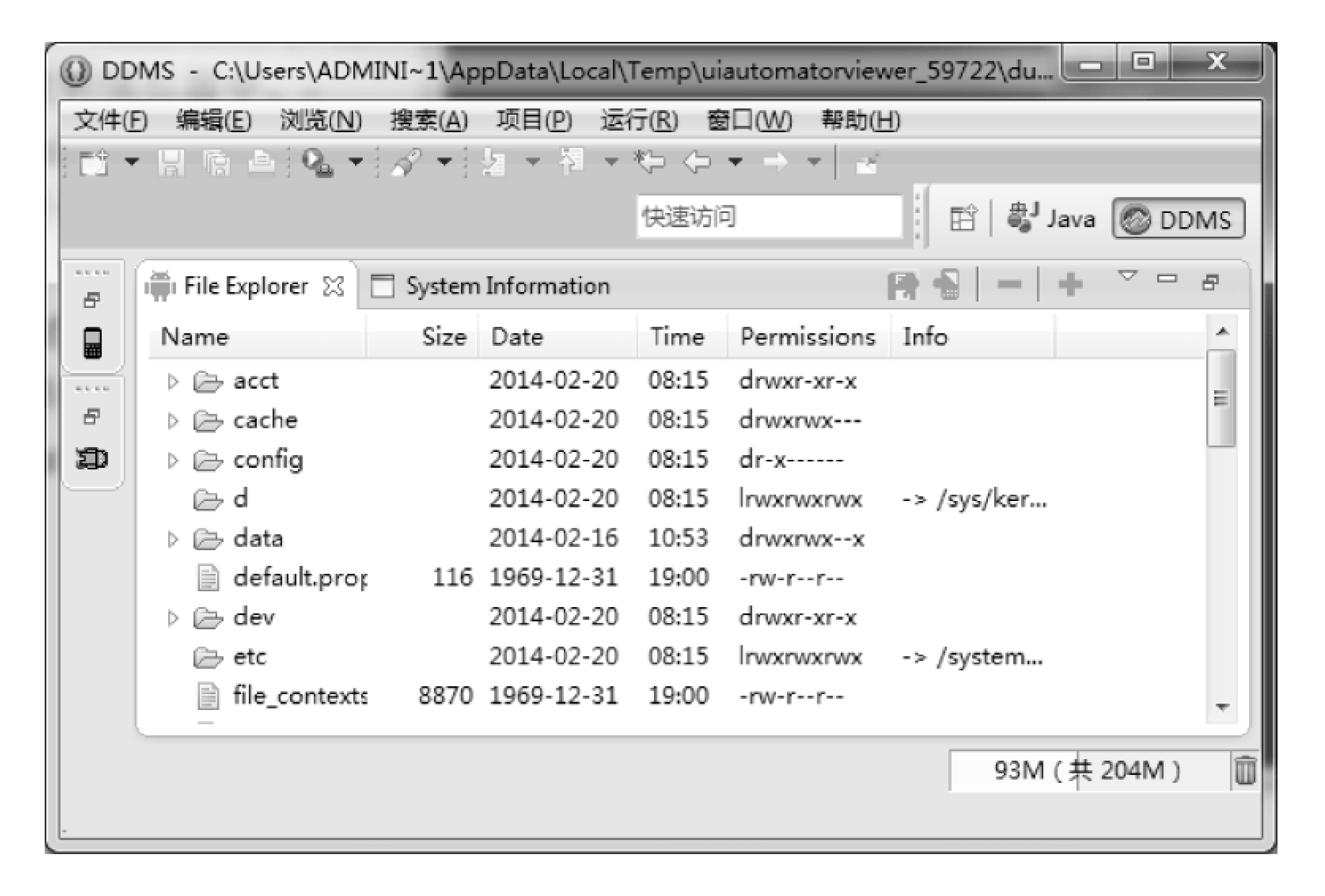


图 1-36　文件目录

上角的操作按钮从左到右分别为 Android 设备保存到本地、上传到 Android 设备、删除文件、添加文件夹。在使用这 4 个功能时，需要对 Android 设备的文件系统具有相应的操作权限。

（4）日志输出（LogCat）：在模拟器中的所有输出信息都显示在日志信息中，该视图一般在最下方，如图 1-37 所示。

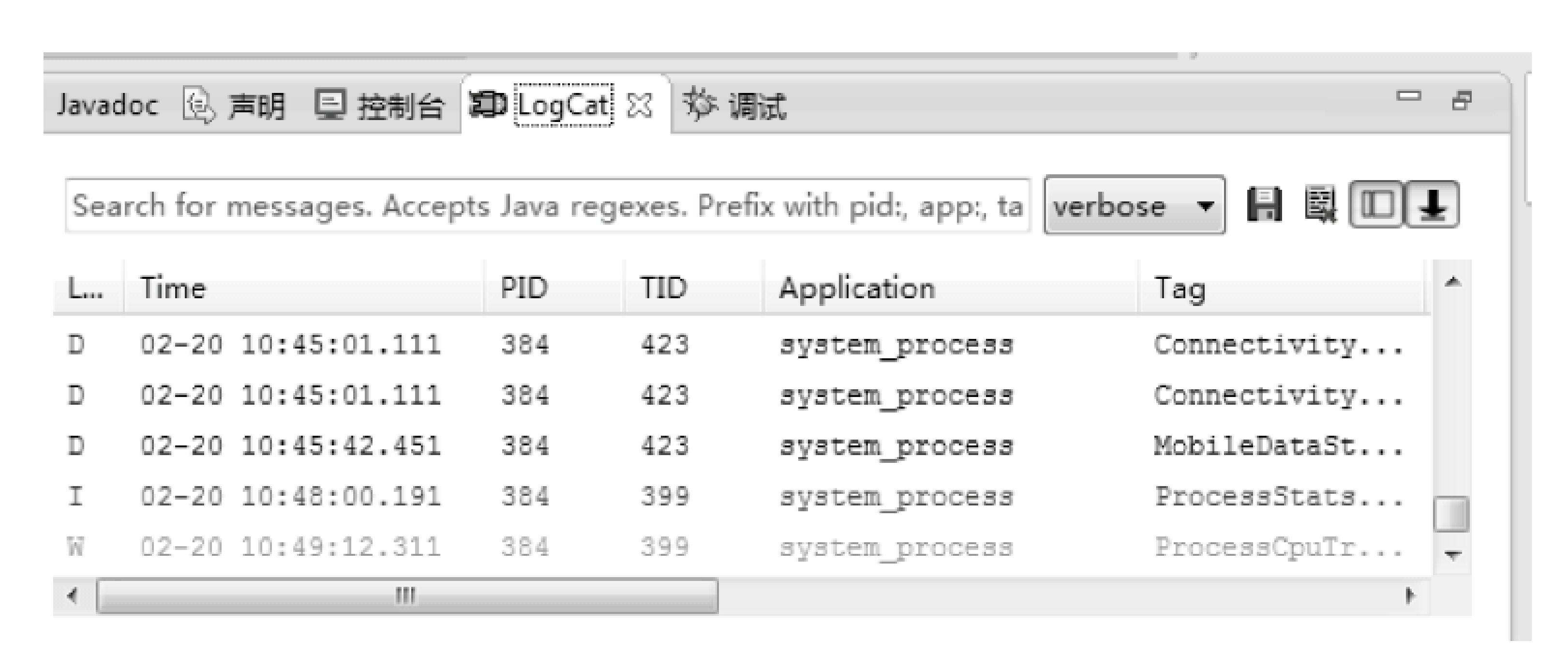


图 1-37　日志信息

在 LogCat 中显示所有测度终端操作的日志记录，通过不同颜色的输出可以很明显地区分警告信息和错误信息，并且可以使用右边的下拉菜单进行不同类型信息的筛选。

1.4.3　Debug 调试

由于 Android 应用程序使用 Java 语言编写，对 Android 应用程序的 Debug 调试和对标准 Java 语言的调试是相同的。

当在工程文件中标记了断点之后，可以使用两种方式开启调试。

- 一种是用右键单击项目，选择 Debug as 项，从应用程序开始运行就开始调试；
- 另一种是在应用程序运行后，在 DDMS 界面的 Devices 选项卡中，使用调试按钮开启调试。

1.5 Android 应用实例

在前面已经学习了怎样在 Eclipse 创建一个 Android 应用项目，下面通过一个应用实例来演示 Android 软件的功能。

【例 1-1】 实现的功能：一个应用可以在屏幕左侧使用一个 fragment 来展示一个文章的标题，然后在屏幕右侧使用另一个 fragment 来展示内容。

其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li1_1fragment。

(2) 打开 res\layout 目录下的 main.xml 文件，其代码为：

```
<!-- 线性布局 -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:baselineAligned="false"
android:orientation="horizontal"
android:background="#99FFCC">                    <!-- 设置背景图 -->
<fragment                                        <!-- 添加 fragment 控件 -->
    android:id="@+id/titles"
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="1"
    class="cn.eoe.leigo.fragmentdemo.TitlesFragment" />
<FrameLayout                                     <!-- 帧布局 -->
    android:id="@+id/details"
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:background="#99FFCC"/>
</LinearLayout>
```

(3) 在 src\fs.li1_1fragment 包下创建一个名为 TitlesFragment.java 文件，首先选中 fs.li1_1fragment 包并右击，在弹出的快捷菜单中选择“新建”→“类”选项，操作步骤如图 1-38 所示。此时弹出“新建 Java 类”文本框，在“包(K)”右侧的文本框中输入对应所要创建的 Java 类，单击“超类(S)”右侧的“浏览(E)”按钮，弹出如图 1-39 所示的“选择超类”文本框，在文本框中的“选择类型(C)”文本框中输入对应的超类名后，单击“确定”按钮，即可完成超类的选择，最后得到如图 1-40 所示的“新建 Java 类”文本框，单击“完成”按钮，即完成 Java 类的创建。

TitlesFragment.java 文件继承了 ListFragment，用来显示标题，其代码为：

```
package fs.li1_1fragment;
import android.annotation.SuppressLint;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.app.ListFragment;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
@SuppressLint("NewApi")
```

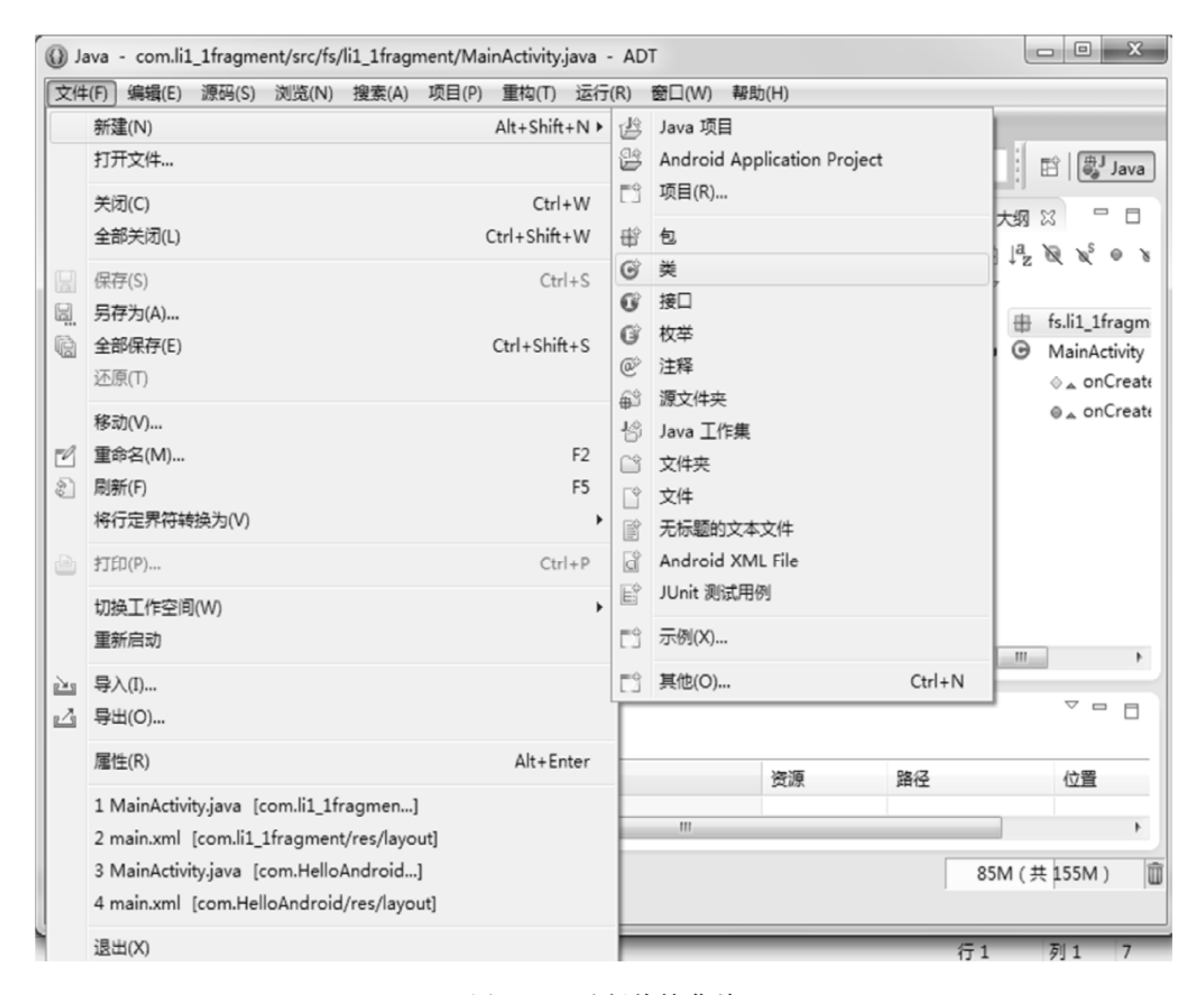


图 1-38　选择快捷菜单

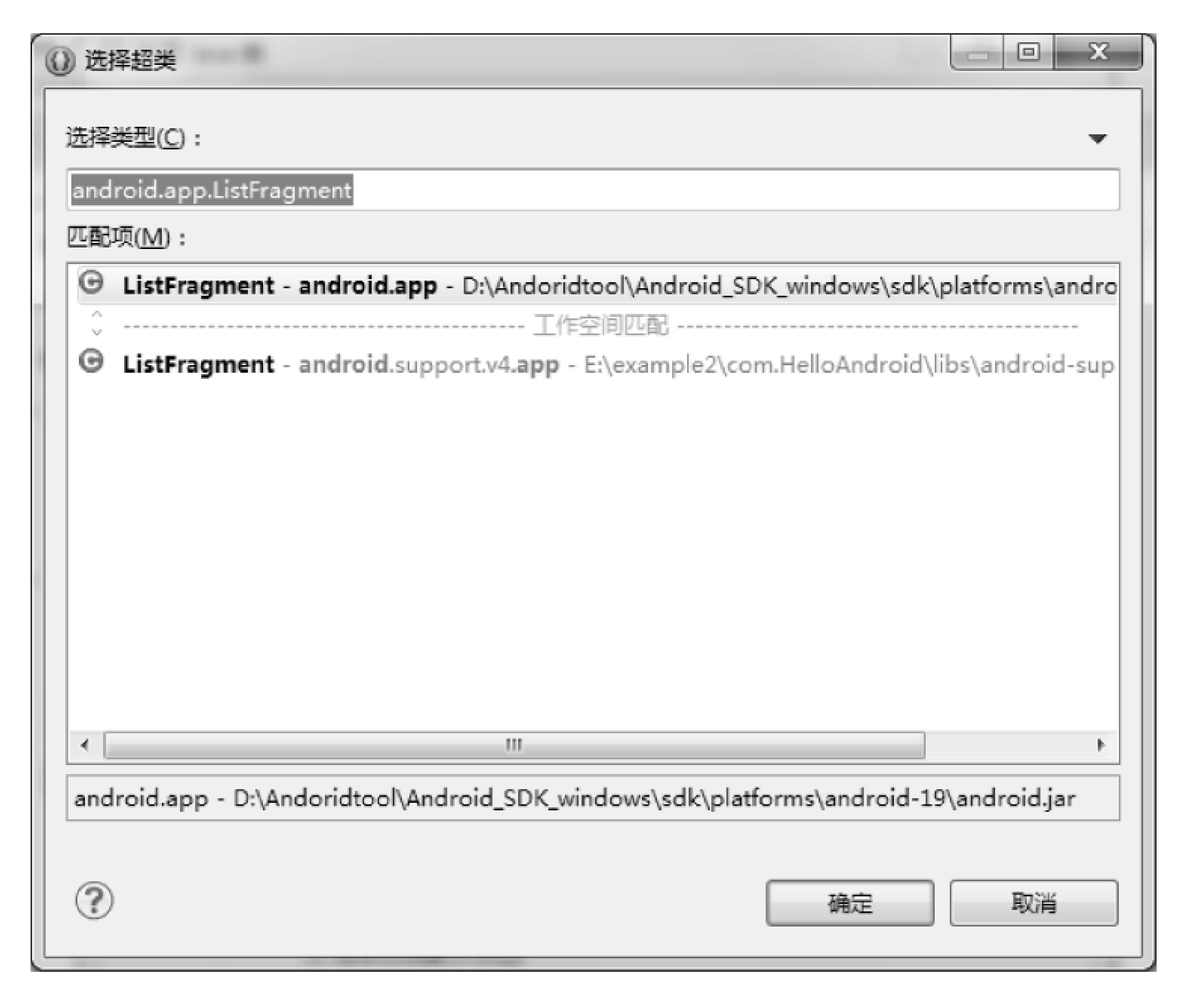


图 1-39　“选择超类”窗口



图 1-40 “新建 Java 类”窗口

```
public class TitlesFragment extends ListFragment {
    private static final String[] WEEKS = new String[] { "Sunday", "Monday","Tuesday",
"Wednesday", "Thursday", "Friday", "Saturday" };
    //初始化的选择位置
    int mCurCheckPosition = 0;
    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        //设置要显示的数据
        setListAdapter(new ArrayAdapter<String>(getActivity(),android.R.layout.simple_list_
item_activated_1,android.R.id.text1, WEEKS));
        showDetails(mCurCheckPosition);
        }
    private void showDetails(int index) {
        //fragment 的管理器
        FragmentManager fm = getFragmentManager();
        MainActivity details = (MainActivity) fm.findFragmentById(R.id.details);if (details == null
 || details.getShowIndex()!= index){
            //设置为单选模式
            getListView().setChoiceMode(ListView.CHOICE_MODE_SINGLE);
            //指定条目被选中
            getListView().setItemChecked(index, true);
            details = MainActivity.newInstance(index);
            //新建 DetailFragment 的实例
```

```
            FragmentTransaction ft = fm.beginTransaction();
            //替换 FrameLayout 为 DetailFragment 类似于事务
            ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
        //将得到的 fragment 替换当前的 viewGroup 内容,add 则不替换会依次累加
            ft.replace(R.id.details, details);
            //提交
            ft.commit();
            }
        }
    @Override
    public void onListItemClick(ListView l, View v, int position, long id){
        super.onListItemClick(l, v, position, id);
        showDetails(position);
        }
}
```

(4) 打开 src\fs.li1_1fragment 包下的 MainActivity.java 文件,该文件用于显示内容,其代码为:

```
package fs.li1_1fragment;
import android.annotation.SuppressLint;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
/**
 * 作为界面的一部分,为 fragment 提供一个 layout
 */
@SuppressLint("NewApi")
public class MainActivity extends Fragment {
    private static final String[] WEEKS = new String[] { "Sunday", "Monday","Tuesday",
"Wednesday", "Thursday", "Friday", "Saturday" };
    public static MainActivity newInstance(int index) {
        MainActivity df = new MainActivity();
        Bundle args = new Bundle();
        args.putInt("index", index);
        df.setArguments(args);
        return df;
        }
    public int getShowIndex() {
        int index = getArguments().getInt("index", 0);
        return index;
        }
    @Override
    public View onCreateView (LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        TextView tv = new TextView(getActivity());
        tv.setText(WEEKS[getShowIndex()]);
        return tv;
        }
}
```

(5) 打开 res\value 目录下的 strings.xml 文件,用于为模拟器添加标题,其代码为:

```
<?xml version = "1.0" encoding = "utf-8"?>
```

```
<resources>
    <string name="app_name">FragmentDemo</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
</resources>
```

运行程序,效果如图 1-41 所示。

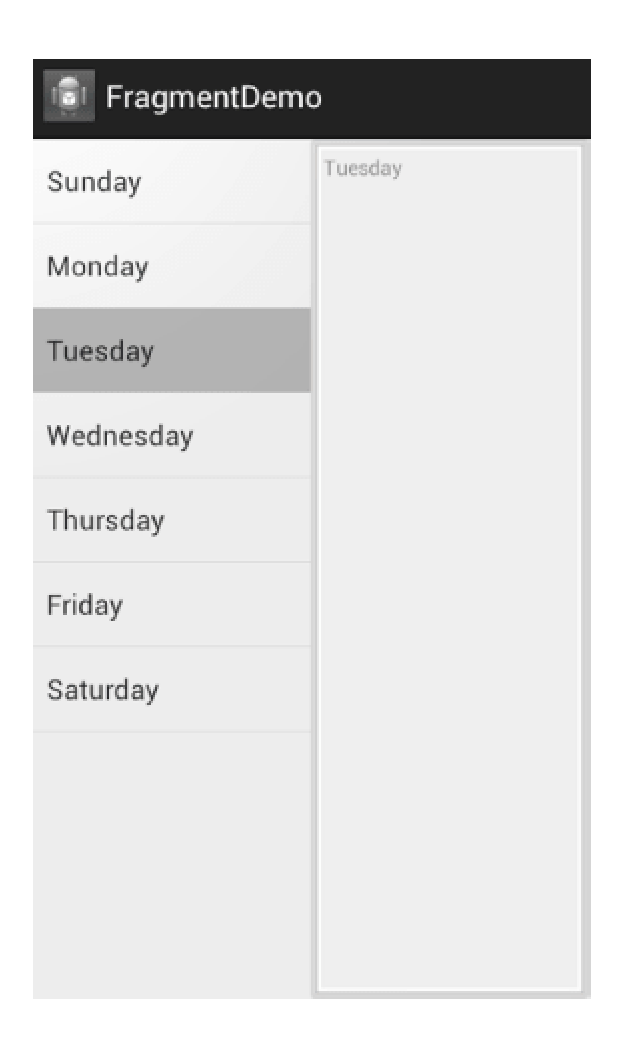


图 1-41　运行效果

至此,可以对 Android 有个大概的认识,而下面将通过实例来介绍经典的 Android 软件。

第2章 Android 布局

手机应用程序相对于一般 PC 应用程序来说，有自己的独特之处，手机分辨率一般为 320×240 或 480×320，这使得界面控件相对有限，要想实现丰富的功能，就必须在开发中灵活使用各种布局(Layout)，并在布局中布置合适的控件去完成程序的功能。这些布局就是使用布局管理器来进行定义的。

Android 程序通常都由几个页面组成，一个这样的页面通常对应一个 XML 文件，而在界面中放置控件前，要确定这个页面是用什么样的布局(布局定义了控件之间的视觉关系)，是采用什么的方式对齐的。当然也可以在一个布局中再嵌套其他布局，控件的对齐方式是以包裹其布局为准的。

View 的布局显示方式包括线性布局(Linear Layout)、相对布局(Relative Layout)、表格布局(Table Layout)、网格视图(Grid View)、标签布局(Tab Layout)、列表视图(List View)、绝对布局(Absolute Layout)。

2.1 View 类概述

2.1.1 View 简介

在介绍 Android 的布局管理器之前，有必要了解 Android 平台下的控件类。首先要了解的是 View 类，该类为所有可视化控件的基类，主要提供了控件绘制和事件处理的方法。创建用户界面所使用的控件都继承自 View，如 TextView、Button、CheckBox 等。

关于 View 及其子类的相关属性，既可以在布局 XML 文件中进行设置，也可以通过成员方法在代码中动态设置。View 类常用的属性及其对应方法如表 2-1 所示。

表 2-1 View 类常用属性及对应方法说明

属性名称	对应方法	描述
android:background	setBackgroundResource(int)	设置背景
android:clickable	setClickable(boolean)	设置 View
android:visibility	setVisibility(int)	控制 View
android:focusable	setFocusable(boolean)	控制 View
android:id	setId(int)	为 View
android:longClickable	setLongClickable(boolean)	设置 View
android:soundEffectsEnabled	setSoundEffectsEnabled(boolean)	设置当 View

续表

属性名称	对应方法	描　　述
android:saveEnabled	setSaveEnabled(boolean)	如果未作设置,当 View
android:nextFocusDown	setNextFocusDownId(int)	定义当向下搜索时应该获取焦点的 View,如果该 View 不存在或不可见,则会抛出 Runtime Exception 异常
android:nextFocusLeft	setNextFocusLeftId(int)	定义当向左搜索时应该获取焦点的 View
android:nextFocusRight	setNextFocusRightId(int)	定义当向右搜索时应该获取焦点的 View
android:nextFocusUp	setNextFocusUpId(int)	定义当向上搜索时应该获取焦点的 View,如果该 View 不存在或不可见,则会抛出 RuntimeException 异常

说明:任何继承自 View 的子类都将拥有 View 类的以上属性及对应方法。

2.1.2　ViewGroup 简介

ViewGroup 类,是 View 类的子类,但是可以充当其他控件的容器。ViewGroup 的子控件既可以是普通的 View,也可以是 ViewGroup,实际上,这是使用了 Composite 的设计模式。Android 中的一些高级控件如 Galley、GridView 等都继承自 ViewGroup。

与 Java SE 不同,Android 中并没有设计布局管理器,而是为每种不同的布局提供了一个 ViewGroup 的子类,常用的布局及其类结构如图 2-1 所示。

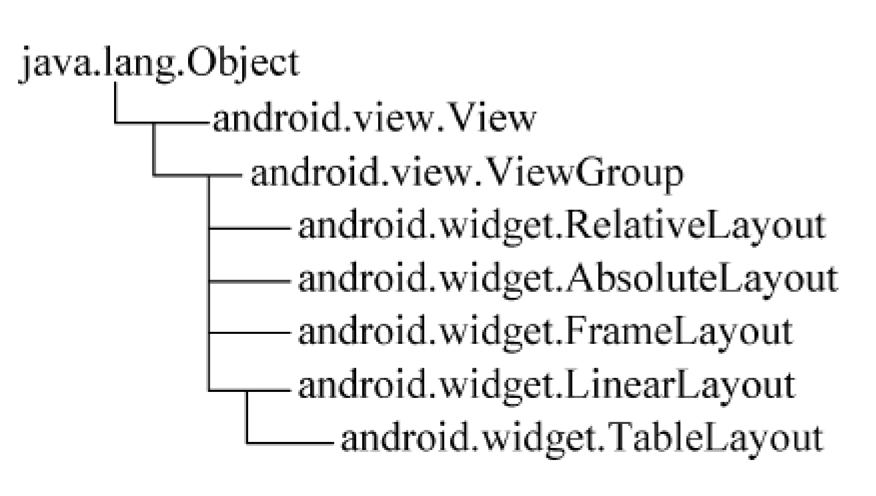


图 2-1　布局管理器的类结构

布局管理器都是以 ViewGroup 为基类派生出来的;使用布局管理器可以适配不同手机屏幕的分辨率、尺寸大小。布局管理器之间的继承关系如图 2-2 所示。

图 2-2 中提到的几个 ViewGroup 如下:

- AbsoluteLayout:绝对定位的布局方式,内部嵌套的元素必须指定具体的位置。
- FrameLayout:帧布局方式,即层布局方式,也就是说,它内部的元素是一层一层的叠加在一起的。如果用过 Photoshop 或 Flash,这里面层的概念是和它们一致的。如果最上层的元素是不透明的,并且比下面的元素尺寸要大,那么将看不到下面的元素,只能看到顶层元素。这些层的顺序是:最新声明的放到最前面。可以这样理解,Android 按文件的书写顺序来组织这个布局,先声明的放在第一层,再声明的放到第二层……最后声明的放在最顶层。
- LinearLayout:线性布局方式,这种布局常用,也简单,就是每个元素占一行,当然也可能声明为横向排放,也就是每个元素占一列。
- RelativeLayout:相对定位的布局方式,在元素的位置,使用相对位置,可以相对其他元素,也可以相对这个布局,就像说:A 现在站在 pawa 和 tempest 的中间;或者说,A 站在队伍的中间。前者就是相对其他元素来定义位置,后者是相对整个布局来定义位置。
- TableLayout:表格的布局方式,这里面的 Table 和 HTML 中的 Table 非常像,就连写法都非常像。

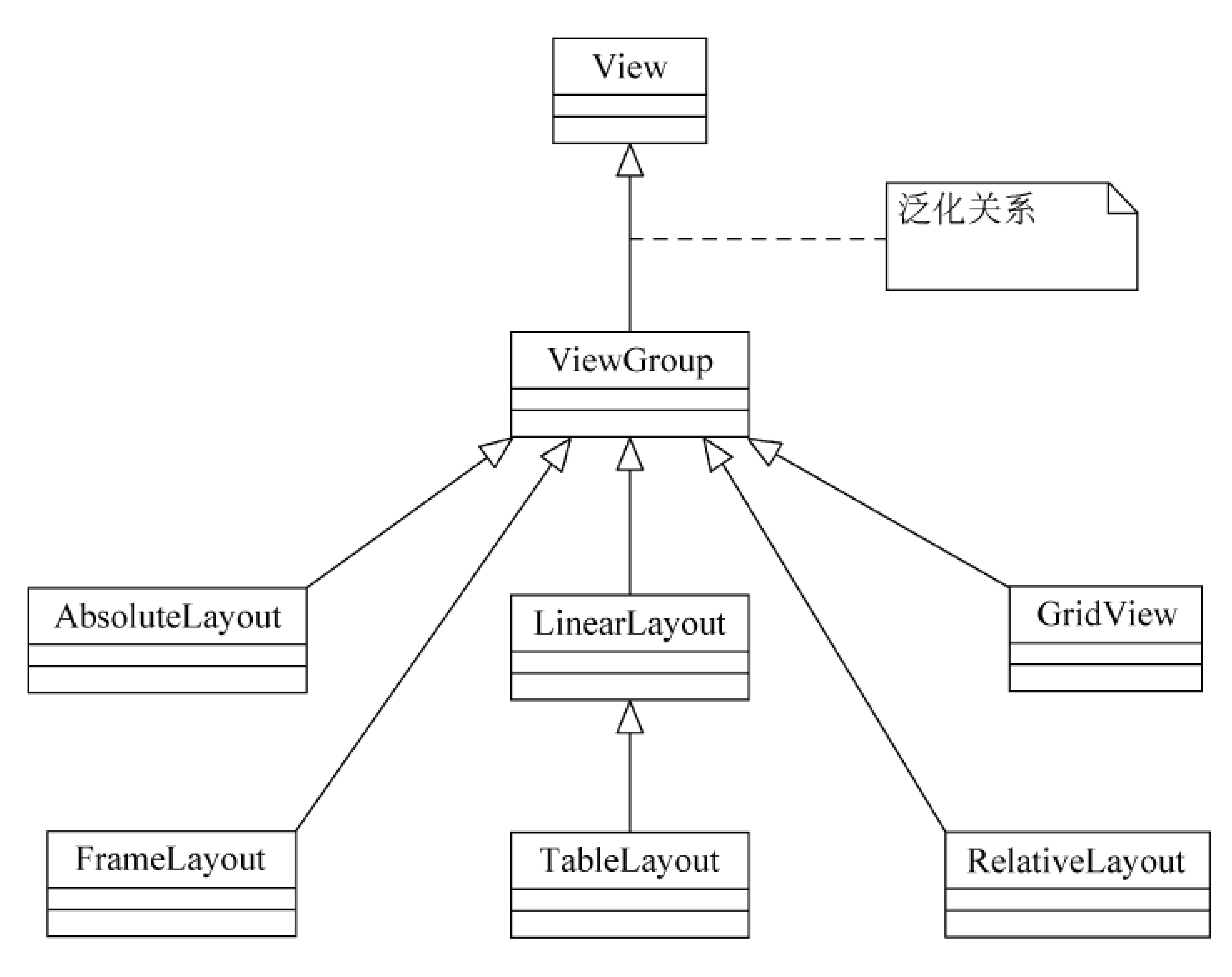


图 2-2 布局器间的继承关系图

2.1.3 自定义 View

虽然 Android 提供了很多继承了 View 类的 UI 组件，但是在实际开发时，还会出现不足以满足程序需要的情况。这时用户就可以通过继承 View 类来开发自己的组件。开发自定义 View 组件的主要步骤如下：

（1）创建一个继承 android.view.View 类的 View 类，并且重写构造方法。

（2）根据需要重写相应的方法。

在代码中右击，在弹出的快捷菜单中选择“代码/覆盖/实现方法”选项，将打开如图 2-3 所示的对话框，在该对话框的列表框中显示了可以被重写的方法。只需选中“被重写的方法”复选框，并单击“确定”按钮，Eclipse 将自动重写指定的方法。一般情况下，不需要重写全部方法。

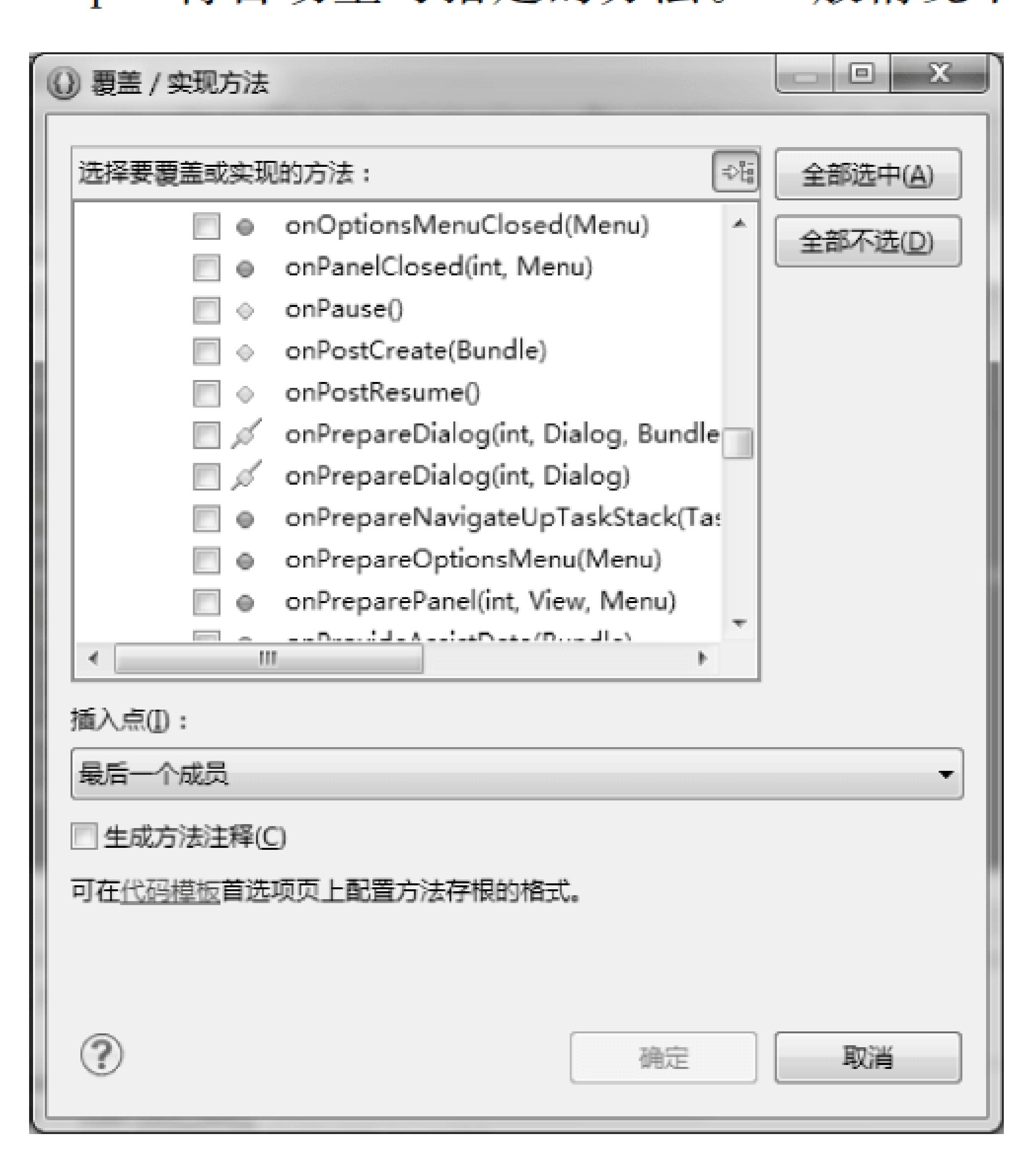


图 2-3 “覆盖/实现方法”对话框

(3) 在项目的活动中,创建并实例化自定义 View 类,并将其添加到布局管理器中。

2.1.4 经典案例

下面通过一个具体的案例来演示怎样开发自定义 View 类。

【例 2-1】 自定义 View 组件实现跟随手指动的小鸭子。

其具体的实现步骤为:

(1) 在 Eclipse 中创建 Android 应用项目,命名为 li2_1View。

(2) 打开 res\layout 目录下的 main.xml 布局文件,其代码修改为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:background = "@drawable/bj1"
    android:id = "@ + id/mylayout" >
</FrameLayout >
```

(3) 创建一个名为 DuckView 的 Java 类,该类继承 android.view.View 类,重写带一个参数 Context 的构造方法和 onDraw()方法。其中,在构造方法中设置小鸭子的默认显示位置,在 onDraw()方法中根据图片绘制小鸭子,其代码为:

```
public class DuckView extends View {
    public float bitmapX;                        //鸭子显示位置的 X 坐标
    public float bitmapY;                        //鸭子显示位置的 Y 坐标
    public DuckView(Context context) {           //重写构造方法
        super(context);
        bitmapX = 750;                           //鸭子的默认显示位置的 X 坐标
        bitmapY = 500;                           //鸭子的默认显示位置的 Y 坐标
    }
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        Paint paint = new Paint();               //创建并实例化 Paint 的对象
        Bitmap bitmap = BitmapFactory.decodeResource(this.getResources(),
                R.drawable.duck1);               //根据图片生成位图对象
        canvas.drawBitmap(bitmap, bitmapX, bitmapY, paint);    //绘制小鸭
        if (bitmap.isRecycled()) {               //判断图片是否回收
            bitmap.recycle();                    //强制回收图片
        }
    }
}
```

(4) 打开主活动文件 MainActivity,它的 onCreate()方法中,首先获取帧布局管理器并实例化小鸭子对象 duck,接着为 duck 添加触摸事件监听器,在重写的触摸事件中设置 duck 的显示位置并重绘 duck 组件,最后将 duck 添加到布局管理器中,其代码为:

```
public class MainActivity extends Activity {
    /** 第一次调用 activity */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获取帧布局管理器
```

```
            FrameLayout frameLayout = (FrameLayout) findViewById(R.id.mylayout);
            //创建并实例化 RabbitView 类
            final DuckView duck = new DuckView(MainActivity.this);
            //为小鸭子添加触摸事件监听
            duck.setOnTouchListener(new OnTouchListener() {
                @Override
                public boolean onTouch(View v, MotionEvent event) {
                    duck.bitmapX = event.getX();          //小鸭子显示位置的 X 坐标
                    duck.bitmapY = event.getY();          //小鸭子显示位置的 Y 坐标
                    duck.invalidate();                    //重绘 duck 组件
                    return true;
                }
            });
            frameLayout.addView(rabbit);                  //将 duck 添加到布局管理器中
        }
    }
```

运行程序，效果如图 2-4(a)所示，当用鼠标在屏幕上拖曳时，小鸭子将跟随鼠标的拖曳轨迹移动，效果如图 2-4(b)所示。




(a) 主界面



(b) 随鼠标拖动的效果图

图 2-4　自定义 View 类

2.2　线性布局

2.2.1　LinearLayout 类概述

LinearLayout 按照垂直或水平的顺序依次排列子元素，每一个子元素都位于前一个元素之后。如果是垂直排列，那么将是一个 N 行单列的结构，每一行只会有一个元素，而不论这个元素的宽度为多少；如果是水平排列，那么将是一个单行 N 列的结构。如果搭建两行两列的结构，通常的方式是先垂直排列两个元素，每一个元素里再包含一个 LinearLayout 进行水平排列。

表 2-2 给出了 LinearLayout 常用的属性及这些属性的对应设置方法。

表 2-2　LinearLayout 常用属性及对应方法

属性名称	对应方法	描　　述
android:orientation	setOrientation(int)	设置线性布局的朝向,可取 horizontal
android:gravity	setGravity(int)	设置线性布局的内部元素的布局方式

在线性布局中可使用 gravity 属性来设置控件的对齐方式,gravity 可取的值及说明如表 2-3 所示。

表 2-3　gravity 可取的属性及说明

属　性　值	描　　述
top	不改变控件大小,对齐到容器顶部
bottom	不改变控件大小,对齐到容器底部
left	不改变控件大小,对齐到容器左侧
right	不改变控件大小,对齐到容器右侧
center_vertical	不改变控件大小,对齐到容器纵向中央位置
center-horizontal	不改变控件大小,对齐到容器横向中央位置
center	不改变控件大小,对齐到容器中央位置
fill_vertical	若有可能,纵向拉伸以填满容器
fill_horizontal	若有可能,横向拉伸以填满容器
fill	若有可能,纵向横向同时拉伸以填满容器

提示:当需要为 gravity 设置多个值时,用|分隔即可。

2.2.2　经典案例

本节通过一个案例来说明 LinearLayout 的用法。

【例 2-2】　利用 LinearLayout 不同的属性来布局按钮的使用。

其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li2_2LinearLayou。

(2) 打开 res\layout 目录下的 main.xml 文件,文件中使用线性布局,声明几个 Button 控件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:orientation = "horizontal"
        android:background = "#99FFCC">
< Button android:id = "@ + id/button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "按钮 1"
        android:layout_weight = "1"/>
< Button android:id = "@ + id/button2"
      android:layout_width = "wrap_content"
      android:layout_height = "wrap_content"
```

```
        android:text = "按钮 2"
        android:layout_weight = "1"/>
    < Button android:id = "@ + id/button3"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "按钮 3"
        android:layout_weight = "1" />
    < Button android:id = "@ + id/button4"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "按钮 4"
        android:layout_weight = "1" />
    < Button android:id = "@ + id/button5"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "按钮 5"
        android:layout_weight = "1"/>
    </LinearLayout >
```

可以看出，根 LinearLayout 视图组（ViewGroup）包含 5 个 Button，它的子元素是以线性方式（horizontal，水平的）布局，运行效果如图 2-5 所示。

如果在 android:orientation＝"horizontal"设置为 vertical，按钮排列则是垂直或者说是纵向的，效果如图 2-6 所示。



图 2-5　按钮横向排列



图 2-6　按钮的纵向排列

2.3　相对布局

2.3.1　RelativeLayout 类概述

RelativeLayout 按照各子元素之间的位置关系完成布局。在此布局中的子元素里与位置相关的属性将生效。例如，android: layout_below、android: layout_above、android: layout_

centerVertical 等。注意，在指定位置关系时，引用的 ID 必须在引用之前，先被定义，否则将出现异常。

RelativeLayout 是 Android 五大布局结构中最灵活的一种布局结构，适合一些复杂界面的布局。

在进行相对布局时用到的属性很多，首先来看属性值只为 true 或 false 的属性，如表 2-4 所示。

表 2-4　相对布局中只取 true 或 false 的属性

属性名称	描　　述
android:layout_centerHorizontal	当前控件位于父控件的横向中间位置
android:layout_centerVertical	当前控件位于父控件的纵向中间位置
android:layout_centerInParent	当前控件位于父控件的中央位置
android:layout_alignParentBottom	当前控件底端与父控件底端对齐
android:layout_alignParentLeft	当前控件左侧与父控件左侧对齐
android:layout_alignParentRight	当前控件右侧与父控件右侧对齐
android:layout_alignParentTop	当前控件顶端与父控件顶端对齐
android:layout_alignWithParentIfMissing	参照控件不存在或不可见时参照父控件

接下来再来看属性值为其他控件 id 的属性，如表 2-5 所示。

表 2-5　相对布局中取值为其他控件 id 的属性及说明

属性名称	描　　述
android:layout_toRightOf	使当前控件位于给出 id 的右侧
android:layout_toLeftOf	使当前控件位于给出 id 的左侧
android:layout_above	使当前控件位于给出 id 的上面
android:layout_below	使当前控件位于给出 id 的下面
android:layout_alignTop	使当前控件的上边界与给出 id 对齐
android:layout_alignBottom	使当前控件的下边界与给出 id 对齐
android:layout_alignLeft	使当前控件的左边界与给出 id 对齐
android:layout_alignRight	使当前控件的右边界与给出 id 对齐

最后要介绍的是属性值以像素为单位的属性及说明，如表 2-6 所示。

表 2-6　相对布局中取值为像素的属性及说明

属性名称	描　　述
android:layout_marginLeft	当前控件左侧的留白
android:layout_marginRight	当前控件右侧的留白
android:layout_marginTop	当前控件上方的留白
android:layout_marginBottom	当前控件下方的留白

2.3.2　经典案例

下面通过一个案例来演示 RelativeLayout 的使用。

【例 2-3】　利用 RelativeLayout 类展示"烟花"布局。

其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li2_3yanhua。

(2) 打开 res\layout 目录下的 main. xml 文件,在文件中使用 RelativeLayout 布局,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
<!-- 定义该组件位于父容器中间 -->
<TextView
    android:id = "@ + id/view01"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:background = "@drawable/leaf"
    android:layout_centerInParent = "true"/>
<!-- 定义该组件位于 view01 组件的上方 -->
<TextView
    android:id = "@ + id/view02"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:background = "@drawable/leaf"
    android:layout_above = "@id/view01"
    android:layout_alignLeft = "@id/view01"/>
<!-- 定义该组件位于 view01 组件的下方 -->
<TextView
    android:id = "@ + id/view03"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:background = "@drawable/leaf"
    android:layout_below = "@id/view01"
    android:layout_alignLeft = "@id/view01"/>
<!-- 定义该组件位于 view01 组件的左边 -->
<TextView
    android:id = "@ + id/view04"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:background = "@drawable/leaf"
    android:layout_toLeftOf = "@id/view01"
    android:layout_alignTop = "@id/view01"/>
<!-- 定义该组件位于 view01 组件的右边 -->
<TextView
    android:id = "@ + id/view05"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:background = "@drawable/leaf"
    android:layout_toRightOf = "@id/view01"
    android:layout_alignTop = "@id/view01"/>
</RelativeLayout>
```

运行程序,效果如图 2-7 所示。



图 2-7　烟花布局效果

2.4 表格布局

2.4.1 TableLayout 类概述

表格布局(TableLayout)适用于 N 行 N 列的布局格式。一个 TableLayout 由许多 TableRow 组成,一个 TableRow 就代表 TableLayout 中的一行。

TableRow 是 LinearLayout 的子类,TableLayout 并不需要明确地声明包含多少行、多少列,而是通过 TableRow,以及其他组件来控制表格的行数和列数,TableRow 也是容器,因此可以向 TableRow 里面添加其他组件,每添加一个组件该表格就增加一列。如果向 TableLayout 里面添加组件,那么该组件就直接占用一行。在表格布局中,列的宽度由该列中最宽的单元格决定,整个表格布局的宽度取决于父容器的宽度(默认是占满父容器本身)。

TableLayout 继承了 LinearLayout,因此它完全可以支持 LinearLayout 所支持的全部 XML 属性,除此之外 TableLayout 还支持如表 2-7 所示的属性。

表 2-7 TableLayout 类常用属性及对应方法说明

属性名称	对应方法	描述
android:collapseColumns	setColumnCollapsed(int,boolean)	设置指定列号的列为 Collapsed
android:shrinkColumns	setShrinkAllColumns(boolean)	设置指定列号的列为 Shrinkable
android:stretchColumns	setStretchAllColumns(boolean)	设置指定列号的列为 Stretchable

说明: setShrinkAllColumns 和 setStretchAllColumns 实现的功能是将表格中的所有列设置为 Shrinkable 或 Stretchable。

2.4.2 经典案例

下面通过一个案例来演示使用 TableLayout 来显示按钮的显示方式。

【例 2-4】 按钮的排列方式。

其具体操作步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li2_4TableLayout。

(2) 打开 res\layout 目录下的 main.xml 文件,使用 TableLayout 布局,其代码为:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:orientation="vertical"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:background="#88FFAA" >
 <!-- 第一个表格布局,指定第 2 列允许收缩,第 3 列允许拉伸 -->
 <TableLayout android:id="@+id/tl1"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:shrinkColumns="1"
 android:stretchColumns="2">
<!-- 独占一行 -->
 <Button android:layout_height="wrap_content"
 android:layout_width="wrap_content"
 android:text="独自一行的按钮"
 android:id="@+id/btt1"/>
```

```
<TableRow>
<Button android:layout_height="wrap_content"
android:layout_width="wrap_content"
android:text="普通按钮"
android:id="@+id/btt2" />
<Button android:layout_height="wrap_content"
android:layout_width="wrap_content"
android:text="允许收缩按钮"
android:id="@+id/btt3"/>
<Button android:layout_height="wrap_content"
android:layout_width="wrap_content"
android:text="可以被允许拉伸按钮"
android:id="@+id/btt4"/>
</TableRow>
</TableLayout>
<!-- 第二个表格布局,指定第 2 列隐藏 -->
<TableLayout android:id="@+id/tl2"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:collapseColumns="1" >
<!-- 独占一行 -->
<Button android:layout_height="wrap_content"
android:layout_width="wrap_content"
android:text="独自一行的按钮"
android:id="@+id/btt5"/>
<TableRow>
<Button android:layout_height="wrap_content"
android:layout_width="wrap_content"
android:text="普通按钮 1"
android:id="@+id/btt6" />
<Button android:layout_height="wrap_content"
android:layout_width="wrap_content"
android:text="普通按钮 3"
android:id="@+id/btt7"/>
<Button android:layout_height="wrap_content"
android:layout_width="wrap_content"
android:text="普通按钮 3"
android:id="@+id/btt8"/>
</TableRow>
</TableLayout>
<!-- 第三个表格布局,指定第 2 和第 3 列允许拉伸 -->
<TableLayout android:id="@+id/tl3"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:stretchColumns="1,2" >
<!-- 独占一行 -->
<Button android:layout_height="wrap_content"
android:layout_width="wrap_content"
android:text="独自一行的按钮"
android:id="@+id/btt9"/>
<TableRow>
<Button android:layout_height="wrap_content"
android:layout_width="wrap_content"
android:text="普通按钮 1"
android:id="@+id/btt10" >
</Button>
<Button android:layout_height="wrap_content"
android:layout_width="wrap_content"
```

```
android:text = "允许拉伸按钮"
android:id = "@ + id/btt11"/>
< Button android:layout_height = "wrap_content"
android:layout_width = "wrap_content"
android:text = "普通按钮 3"
android:id = "@ + id/btt12"/>
</TableRow >
< TableRow >
< Button android:layout_height = "wrap_content"
android:layout_width = "wrap_content"
android:text = "允许拉伸按钮"
android:id = "@ + id/btt13" >
</Button >
< Button android:layout_height = "wrap_content"
android:layout_width = "wrap_content"
android:text = "允许拉伸按钮"
android:id = "@ + id/btt14"/>
</TableRow >
</TableLayout >
</LinearLayout >
```

运行程序,效果如图 2-8 所示。



图 2-8　按钮的排列效果

2.5　切　换　卡

TabWidget 类概述如下:

切换卡(TabWidget)是一种相对复杂的布局管理器,通过多个标签来切换显示不同的内容,一个 TabWidget 主要是由一个 TabHost 来存放多个 Tab 标签容器,再在 Tab 容器中加入其他控件,通过 addTab 方法可以增加新的 Tab,这些除了在 XML 文件中布局好控制外,当然还需要在 Java 文件中处理好事件的逻辑。

TabWidget 继承自 LinearLayout,是线性布局的一种,除了继承自父类的属性和方法,在 FrameLayout 类中包含了自己特有的属性和方法,如表 2-8 所示。

表 2-8　TabWidget 常用的属性

属　　性	描　　述
android:divider	可绘制对象,被绘制在选项卡窗口间充当分割物
android:tabStripEnabled	确定是否在选项卡绘制
android:tabStripLeft	用来绘制选项卡下面的分割线左边部分的可视化对象
android:tabStripRight	用来绘制选项卡下面的分割线右边部分的可视化对象

2.6　经 典 案 例

下面通过一个案例来演示 TabWidget 类的使用。

【例 2-5】　使用 TabWidget 实现切换卡。

其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li2_5TabWidget。

(2) 打开 res\Layout 目录下的 main. xml 文件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<!-- 定义了 TabHost 布局,其 id 必须为 @android:id/tabhost -->
<TabHost xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@android:id/tabhost"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
    <!-- 定义了 3 个切换卡的整体布局方式 -->
    <LinearLayout
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent">
         <!-- 定义了切换卡 TabWidget,其 id 必须为@android:id/tabs -->
        <TabWidget
            android:id = "@android:id/tabs"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content" />
        <!-- 定义了切换卡内 FrameLayout 布局,其 id 必须为@android:id/tabcontent -->
        <FrameLayout
            android:id = "@android:id/tabcontent"
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent">
            <TextView
                android:id = "@ + id/textview1"
                android:layout_width = "fill_parent"
                android:layout_height = "fill_parent"
                android:text = "这是一个 tab" />
                <TextView
                    android:id = "@ + id/textview2"
                    android:layout_width = "fill_parent"
                    android:layout_height = "fill_parent"
                    android:text = "这是另一个 tab" />
                 <TextView
                    android:id = "@ + id/textview3"
                    android:layout_width = "fill_parent"
                    android:layout_height = "fill_parent"
                    android:text = "这是第三个 tab" />
         </FrameLayout>
      </LinearLayout>
</TabHost>
```

(3) 打开 src\fs.li2_5tabwidget 目录下的 MainActivity 文件,其代码为:

```
public class MainActivity extends TabActivity
{
    //声明 TabHost 对象
    TabHost mTabHost;
    /** 第一次调用 activity  */
        @Override
        public void onCreate(Bundle savedInstanceState)
        {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //取得 TabHost 对象
        mTabHost = getTabHost();
        /* 为 TabHost 添加标签 */
        //新建一个 newTabSpec(newTabSpec)
        //设置其标签和图标(setIndicator)
        //设置内容(setContent)
```

```
        mTabHost.addTab(mTabHost.newTabSpec("tab_test1")
            .setIndicator("TAB 1",getResources().getDrawable(R.drawable.b))
            .setContent(R.id.textview1));
        mTabHost.addTab(mTabHost.newTabSpec("tab_test2")
            .setIndicator("TAB 2",getResources().getDrawable(R.drawable.b1))
            .setContent(R.id.textview2));
        mTabHost.addTab(mTabHost.newTabSpec("tab_test3")
            .setIndicator("TAB 3",getResources().getDrawable(R.drawable.b2))
            .setContent(R.id.textview3));
        //设置 TabHost 的背景颜色
        mTabHost.setBackgroundColor(Color.argb(150, 22, 70, 150));
        //设置 TabHost 的背景图片资源
        mTabHost.setBackgroundResource(R.drawable.bj1);
        //设置当前显示哪一个标签
        mTabHost.setCurrentTab(0);
        //标签切换事件处理,setOnTabChangedListener
        mTabHost.setOnTabChangedListener(new OnTabChangeListener()
        {
            @Override
            public void onTabChanged(String tabId)
            {
                Dialog dialog = new AlertDialog.Builder(MainActivity.this)
                    .setTitle("提示")
                    .setMessage("当前选中: " + tabId + "标签")
                    .setPositiveButton("确定", new DialogInterface.OnClickListener()
                    {
                    public void onClick(DialogInterface dialog, int whichButton)
                    {
                        dialog.cancel();
                    }
                }).create();        //创建按钮
                dialog.show();
            }
        });
    } }
```

运行程序,效果如图 2-9 所示。



图 2-9　切换卡界面

2.7 绝对布局

2.7.1 AbsoluteLayout 类概述

AbsoluteLayout，顾名思义，就是绝对位置的布局；也可以叫做坐标布局，是指定元素的绝对位置（或叫绝对坐标值）。这种布局简单直接，直观性强，但是由于手机屏幕尺寸差别比较大，使用绝对定位的适应性会比较差。

在此布局中子元素的 android:layout_x 和 android:layout_y 属性将生效，用于描述该子元素的坐标位置。屏幕左上角为坐标原点（0,0），第一个 0 代表横坐标，向右移动此值增大，第二个 0 代表纵坐标，向下移动，此值增大。在此布局中的子元素可以相互重叠。在实际开发中，通常不采用此布局格式，因为它的界面代码过于刚性，以至于不能很好的适配各种终端。

2.7.2 经典案例

下面通过一个案例来演示 AbsoluteLayout 类的使用。

【例 2-6】 利用 AbsoluteLayout 类实现一个登录界面。

其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li2_6AbsoluteLayout。

(2) 打开 res\layout 目录下的 main.xml 文件，使用绝对布局，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<!-- android:padding = "10dip"语句为为部分界面添加颜色 -->
< AbsoluteLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:padding = "10dip"
    android:background = "#55FF66">
  < TextView android:id = "@ + id/lable"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:text = "请输入用户名: "/>
  < EditText android:id = "@ + id/text"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:layout_x = "100dip"
            android:layout_y = "20dip"/>
  < Button android:id = "@ + id/cancel"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content"
          android:layout_x = "10dip"
          android:layout_y = "50dip"
          android:text = "取消"/>
  < Button android:id = "@ + id/ok"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content"
          android:layout_x = "60dip"
          android:layout_y = "50dip"
          android:text = "确定"/>
</AbsoluteLayout >
```

运行程序，效果如图 2-10 所示。



图 2-10　登录界面

2.8　帧　布　局

2.8.1　FrameLayout 类概述

FrameLayout 是五大布局中最简单的一个布局，可以说成是层布局方式。在这个布局中，整个界面被当成一块空白备用区域，所有的子元素都不能被指定放置的位置，统统放于这块区域的左上角，并且后面的子元素直接覆盖在前面的子元素之上，将前面的子元素部分和全部遮挡，由子控件来决定。如果子控件一样大，同一时刻只能看到最上面的子控件。

FrameLayout 继承自 ViewGroup，除了继承自父类的属性和方法，FrameLayout 类中包含了自己特有的属性和方法，如表 2-9 所示。

表 2-9　FrameLayout 属性及对应方法

属性名称	对应方法	描　述
android:foreground	setForeground(Drawable)	设置绘制在所有子控件之上的内容
android:foregroundGravity	setForegroundGravity(int)	设置绘制在所有子控件之上内容的 gravity

提示：在 FrameLayout 中，子控件是通过栈来绘制的，所以后添加的子控件会被绘制在上层。

2.8.2　经典案例

下面通过一个案例来演示 FrameLayout 类的用法。

【例 2-7】 利用帧布局实现渐进效果。

其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li2_7FrameLayout。

(2) 打开 res\layout 目录下的 main.xml 文件，利用帧布局实现渐进效果，其代码为：

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:background="#000000">
<!-- 一次定义 7 个 Textview,先定义的 TextView 位于底层,后定义的 TextView 位于上层 -->
<TextView android:id="@+id/view01"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:width="210px"
  android:height="50px"
  android:background="#ff0000"/>
<TextView android:id="@+id/view02"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:width="180px"
  android:height="50px"
  android:background="#dd0000"/>
<TextView android:id="@+id/view03"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:width="150px"
  android:height="50px"
  android:background="#bb0000"/>
<TextView android:id="@+id/view04"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:width="120px"
  android:height="50px"
  android:background="#990000"/>
<TextView android:id="@+id/view05"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:width="90px"
  android:height="50px"
  android:background="#770000"/>
<TextView android:id="@+id/view06"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:width="60px"
  android:height="50px"
  android:background="#550000"/>
<TextView android:id="@+id/view07"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:width="30px"
  android:height="50px"
  android:background="#330000"/>
</FrameLayout>
```

运行程序，效果如图 2-11 所示。



图 2-11　渐进效果

2.9　Tab　容　器

2.9.1　TabHost 类的概述

TabHost 是一种非常实用的组件，TabHost 可以很方便地在窗口上放置多个选项卡，每个选项卡相当于获得了一个与外部容器相同大小的组件摆放区域。通过这种方式，就可以在一个容器里放置更多组件。与 TabHost 结合使用的还有如下组件：

- TabWidget：代表选项卡的标签。
- TabSpec：代表选项卡的一个 Tab 页面。

TabHost 仅仅是一个简单的容器，提供了如下两个方法来创建和添加选项卡：

- newTabSpec(String tag)：创建选项卡。
- addTab(TabHost.TabSpec tabSpec)：添加选项卡。

使用 TabHost 的一般步骤如下：

(1) 在界面布局中定义 TabHost 组件，并为该组件定义该选项卡的内容。

(2) Activity 应该继承 TabActivity。

(3) 调用 TabActivity 的 getTabHost()方法获取 TabHost 对象。

(4) 通过 TabHost 对象的方法来创建和添加选项卡。

TabHost 容器内部需要组合两个组件，分别为 TabWidget 和 FrameLayout。其中，TabWidget 用于定义选项卡的标题；FrameLayout 则用于“层叠”组合多个选项页面。

2.9.2 经典案例

下面通过一个案例来演示 TabHost 类的用法。

【例 2-8】 利用 TabHost 类在屏幕上实现多个选项卡。

其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Andriod 应用项目，命名为 li2_8TabHost。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中布局一个 TabHost 容器，并在窗口中布局三个标签，其代码为：

```
<TabHost xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@android:id/tabhost"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent">
  <RelativeLayout
      android:layout_width = "fill_parent"
      android:layout_height = "fill_parent">
  </RelativeLayout>
  <LinearLayout
      android:layout_width = "match_parent"
      android:layout_height = "match_parent"
      android:orientation = "vertical" >
    <RelativeLayout
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent">
     <FrameLayout
      android:id = "@android:id/tabcontent"
      android:layout_width = "match_parent"
      android:layout_height = "match_parent">
      <!-- 定义第一个选项卡的内容 -->
      <LinearLayout
          android:id = "@ + id/tab01"
          android:orientation = "vertical"
          android:layout_width = "fill_parent"
          android:layout_height = "fill_parent">
          <TextView
              android:layout_width = "fill_parent"
              android:layout_height = "wrap_content"
              android:text = "Android 将军"/>
          <TextView
              android:layout_width = "fill_parent"
              android:layout_height = "wrap_content"
              android:text = "Android 软件经典再现"/>
      </LinearLayout>
      <!-- 定义第二个选项卡的内容 -->
      <LinearLayout
          android:id = "@ + id/tab02"
          android:orientation = "vertical"
          android:layout_width = "fill_parent"
          android:layout_height = "fill_parent">
          <TextView
              android:layout_width = "fill_parent"
              android:layout_height = "wrap_content"
              android:text = "Android 将军 2"/>
```

```
            <TextView
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"
                android:text = "Android 软件经典再现 2" />
        </LinearLayout>
          <!-- 定义第三个选项卡的内容 -->
        <LinearLayout
            android:id = "@ + id/tab03"
            android:orientation = "vertical"
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent">
            <TextView
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"
                android:text = "Android 软件 3"/>
            <TextView
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"
                android:text = "Android 软件经典再现" />
        </LinearLayout>
    </FrameLayout>
      <TabWidget
        android:id = "@android:id/tabs"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:layout_alignParentBottom = "true"/>
  </RelativeLayout>
 </LinearLayout>
</TabHost>
```

(3) 打开 src\fs.li2_8tabhost 包下的 MainActivity.java 文件,在文件中实现当选中某个标签时,即在 TextView 显示对应的内容,其代码为:

```
package fs.li2_8tabhost;
import android.os.Bundle;
import android.app.Activity;
import android.app.TabActivity;
import android.view.Menu;
import android.widget.TabHost;
import android.widget.TabHost.TabSpec;
public class MainActivity extends TabActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //获取该 Activity 里面的 TabHost 组件
        TabHost tabHost = getTabHost();
        //创建第一个 Tab 页
        TabSpec tab1 = tabHost.newTabSpec("tab1")
                .setIndicator("Android 软件 1")
                .setContent(R.id.tab01);
        //添加第一个选项卡
        tabHost.addTab(tab1);
        TabSpec tab2 = tabHost.newTabSpec("tab2")
            .setIndicator("Android 软件 2",getResources().getDrawable(R.drawable.ic_launcher))
            .setContent(R.id.tab02);
```

```
            //添加第二个选项卡
            tabHost.addTab(tab2);
            TabSpec tab3 = tabHost.newTabSpec("tab3").setIndicator("Android 软件 3")
                    .setContent(R.id.tab03);
            //添加第三个选项卡
            tabHost.addTab(tab3);
        }
    }
```

运行效果如图 2-12 所示。



图 2-12　实现多个选项卡

2.10　网格布局

2.10.1　GridLayout 类概述

网格布局是 Android 4.0 才有的，在低版本使用该布局需要导入对应支撑库。GridLayout 将整个容器划分成 rows×columns 个网格，每个网格可以放置一个组件，还可以设置一个组件横跨多少列、多少行。不存在一个网格放多个组件情况。

GridLayout 的布局策略简单分为以下三个部分：

首先，它与 LinearLayout 布局一样，也分为水平和垂直两种方式，默认是水平布局，一个控件挨着一个控件从左到右依次排列，但是通过指定 android:columnCount 设置列数的属性后，控件会自动换行进行排列。另外，对于 GridLayout 布局中的子控件，默认按照 wrap_content 的方式设置其显示，这只需在 GridLayout 布局中显式声明即可。

其次，若要指定某控件显示在固定的行或列，只需设置该子控件的 android:layout_row 和 android:layout_column 属性即可。需要注意的是，android:layout_row="0"表示从第一行开

始，android:layout_column="0"表示从第一列开始，这与编程语言中一维数组的赋值情况类似。

最后，如果需要设置某控件跨越多行或多列，只需将该子控件的 android:layout_rowSpan 或 layout_columnSpan 属性设置为数值，再设置其 layout_gravity 属性为 fill 即可，前一个设置表明该控件跨越的行数或列数，后一个设置表明该控件填满所跨越的整行或整列。

2.10.2 经典案例

下面通过一个案例来演示 GridLayout 类的使用。

【例 2-9】 利用 GridLayout 布局编写的简易计算器。

其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li2_9GridLayout。

(2) 打开 res\layout 目录下的 main.xml 文件，用于在屏幕中布局计算器格局，其代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal"
        android:useDefaultMargins="true"
        android:alignmentMode="alignBounds"
        android:columnOrderPreserved="false"
        android:rowCount="5"
        android:columnCount="4"
        android:background="#000000">
    <Button
        android:id="@+id/one"
        android:text="1"/>
    <Button
        android:id="@+id/two"
        android:text="2"/>
    <Button
        android:id="@+id/three"
        android:text="3"/>
    <Button
        android:id="@+id/devide"
        android:text="/"/>
    <Button
        android:id="@+id/four"
        android:text="4"/>
    <Button
        android:id="@+id/five"
        android:text="5"/>
    <Button
        android:id="@+id/six"
        android:text="6"/>
    <Button
        android:id="@+id/multiply"
        android:text="×"/>
    <Button
        android:id="@+id/seven"
        android:text="7"/>
```

```
    <Button
        android:id = "@ + id/eight"
        android:text = "8"/>
    <Button
        android:id = "@ + id/nine"
        android:text = "9"/>
    <Button
        android:id = "@ + id/minus"
        android:text = " - "/>
    <Button
        android:id = "@ + id/zero"
        android:layout_columnSpan = "2"
        android:layout_gravity = "fill"
        android:text = "0"/>
    <Button
        android:id = "@ + id/point"
        android:text = "."/>
    <Button
        android:id = "@ + id/plus"
        android:layout_rowSpan = "2"
        android:layout_gravity = "fill"
        android:text = " + "/>
    <Button
        android:id = "@ + id/equal"
        android:layout_columnSpan = "3"
        android:layout_gravity = "fill"
        android:text = " = "/>
</GridLayout>
```

运行效果如图 2-13 所示。



图 2-13 计算器界面

2.11 布局综合经典案例

前面已经对各种布局进行简要概述并都给出相应的案例及说明。下面再通过一个案例来总结 Android 布局的用法。

【例 2-10】 在 Android 中实现九宫格布局。

其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 aylotexample。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在此使用名为 GridView 的表格布局，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<GridView xmlns:android = "http://schemas.android.com/apk/res/android"
        android:id = "@ + id/GridView"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:numColumns = "auto_fit"
        android:columnWidth = "90dp"
        android:stretchMode = "columnWidth"
        android:gravity = "center"
        android:background = " # aabbcc">
</GridView>
```

在以上代码中需要关注的属性是 columnWidth，这里指定了列的宽度，一个列对象对应一个“可重复的子项”，这个子项就是图片项和图片下方文字显示的部分。如果不指定这个宽度，默认为每行(展示的行，界面)仅仅只显示一个“可重复的子项”，而当指定宽度时(此处指定为 90dp)，如果每行实际行尺寸大于指定宽度，它就会继续将下一个的“可重复的子项”，放置在本行。于是，就呈现一行显示多个子项的情况。numColumns 属性，指定一个自动填充的值，指示了自动填充行。

(3) 接下来需要再创建一个 XML 布局文件，这里写需要“被迭代”的子项(RelativeLayout)。在 res\layout 目录下创建一个 item. xml 文件的方法为，首先选中 layout 文件夹并右击，在弹出的快捷菜单中选择“新建”→“文件”选项，弹出如图 2-14 所示的“新建文件”对话框，在“文件名(M)”文本框中输入对应的文件名，最后单击“确定”按钮，即可完成 item. xml 文件的创建。



图 2-14　创建新文件窗口

item. xml 文件的代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
android:layout_width = "fill_parent"
android:layout_height = "wrap_content"
android:background = "#aabbcc">
< ImageView
    android:layout_width = "wrap_content"
    android:id = "@ + id/ItemImage"
    android:layout_height = "wrap_content"
```

```
        android:layout_centerHorizontal = "true"/>
<TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_below = "@ + id/ItemImage"
        android:id = "@ + id/ItemText"
        android:layout_centerHorizontal = "true" />
</RelativeLayout>
```

(4) 打开 src\ayoutexample 包下的 MainActivity.java 文件，在文件中首先调用 Activity 活动，然后构建 ArrayList 作为数据源，再构建 SimpleAdapter 作为数据适配器，为 gridView 指定适配器对象，其代码为：

```
package fs.ayoutexmaple;
import java.util.ArrayList;
import java.util.HashMap;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.GridView;
import android.widget.SimpleAdapter;
import android.widget.Toast;
public class MainActivity extends Activity {
        /** 第一次调用活动 */
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            GridView gridview = (GridView) findViewById(R.id.GridView);
            ArrayList< HashMap < String, Object >> meumList = new ArrayList < HashMap < String, Object
>>();
            for(int i = 1;i < 10;i++)
            {
                HashMap < String, Object > map = new HashMap < String, Object >();
                map.put("ItemImage", R.drawable.ic_launcher);
                map.put("ItemText", "" + i);
                meumList.add(map);
            }
            SimpleAdapter saItem = new SimpleAdapter(this,
                        meumList,                                    //数据源
                        R.layout.item,                               //xml 实现
                        new String[]{"ItemImage","ItemText"},        //对应 map 的 Key
                       new int[]{R.id.ItemImage,R.id.ItemText});     //对应 R 的 Id
                    //添加 Item 到网格中
                    gridview.setAdapter(saItem);
                    //添加点击事件
                    gridview.setOnItemClickListener(
                        new OnItemClickListener()
                        {
            public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,long arg3)
                            {
                                int index = arg2 + 1;          //id 是从 0 开始的，所以需要 + 1
            Toast.makeText(getApplicationContext(), "你按下了选项：" + index, 0).show();
                                                    //Toast 用于向用户显示一些帮助/提示
```

```
                        }
                    }
                );
        }
}
```

运行效果如图 2-15 所示。



图 2-15　九宫格布局

第 3 章　Android 控件

Android 控件是 Android 布局美丽的 UI 界面最基本也是最重要的组成部分，下面介绍 Android 控件。

3.1　文本类控件

文本类控件主要包括 TextView 控件、EditText 控件以及 AutoCompleteTextView 控件。

3.1.1　文本框控件

1. TextView 控件概述

TextView 继承 View 类，TextView 控件的功能是向用户显示文本内容，同时可选择让用户编辑文本。从功能上来讲，一个 TextView 就是一个完整的文本编辑器，只不过其本身设置为不允许编辑，其子类 EditText 设置为允许用户对内容进行编辑。

TextView 提供了大量的 XML 属性，这些 XML 属性大部分既可适用于 TextView，又可适用于 EditText，但有少量 XML 只能适用于其中之一。表 3-1 显示了 TextView 支持的 XML 属性及说明。

表 3-1　TextView 的 XML 属性及说明

XML 属性	相关方法	描述
android:autoLink	setAutoLinkMask(int)	是否将符合指定格式的文本转换为可单击的超链接形式
android:cursorVisible	setCursorVisible(Boolean)	设置该文本框的光标是否可见
android:drawableBottom	setCompoundDrawables-WithIntrinsicBounds(Drawable,Drawable,Drawable,Drawable)	在文本框内文本的底端绘制指定图像
android:drawableLeft	setCompoundDrawables-WithIntrinsicBounds(Drawable,Drawable,Drawable,Drawable)	在文本框内文本的左端绘制指定图像
android:drawablePadding	setCompoundDrawables-WithIntrinsicBounds(Drawable,Drawable,Drawable,Drawable)	设置文本框内文本与图像的距离

续表

XML 属性	相 关 方 法	描　　述
android:drawableRight	setCompoundDrawables-WithIntrinsicBounds(Drawable,Drawable,Drawable,Drawable)	在文本框内文本的右端绘制指定图像
android:drawableTop	setCompoundDrawables-WithIntrinsicBounds(Drawable,Drawable,Drawable,Drawable)	在文本框内文本的顶端绘制指定图像
android:editable		设置该文本是否允许编辑
android:ellipsize		设置当显示的文本超过了 TextView 的长度时如何处理文本内容
android:gravity	setGravity(int)	设置文本框内文本的对齐方式
android:height	setHeight(int)	设置该文本框的高度(单位为 pixel)
android:hint	setHint(int)	设置当该文本档内容为空时,文本框内默认显示提示文本
android:minHeight	setMinHeight(int)	指定该文本框的最小高度
android:maxHeight	setMaxHeight(int)	指定该文本框的最大高度
android:minWidth	setMinWidth(int)	指定该文本框的最小宽度
android:maxWidth	setMaxWidth(int)	指定该文本框的最大宽度
android:lines	setlines(int)	设置该文本框默认占几行
android:MinLines	setMinLines(int)	设置该文本框最少占几行
android:MaxLines	setMaxLines(int)	设置该文本框最多占几行
android:password	setTransformationMethod(TransformationMethod)	设置文本框为一个密码框,以点代替字符
android:phoneNumber	setKeyListener(KeyListener)	设置该文本框只能接受电话号码
android:scrollHorizontally	setHorizontallyScrolling(Boolean)	设置文本框不够显示全部内容时是否允许水平滚动
android:selectAllOnFocus	setSelectAllOnFocus(Boolean)	如果文本框的内容可选,设置当它获得焦点时是否自动选中所有文本
android:singleLine	setTransformationMethod	设置文本框是否为单行模式,如设为 true,则不会换行
android:shadowColor	setShadowLayer(float,float,float,int)	设置文本框内文本的阴影颜色
android:shadowDx	setShadowLayer(float,float,float,int)	设置文本框内文本的阴影在水平方向的偏移
android:shadowDy	setShadowLayer(float,float,float,int)	设置文本框内文本的阴影在垂直方向的偏移
android:shadowRadius	setShadowLayer(float,float,float,int)	设置文本框内文本的阴影角度
android:text	setText(CharSequence)	设置文本框内文本的内容
android:textColor	setTextColor(ColorStateList)	设置文本框内文本的颜色

续表

XML 属性	相关方法	描　述
android:textColorHighlight	setHighlightColor(int)	设置文本框内文本被选中时颜色
android:textScaleX	setTextScaleX(float)	设置文本框内文本在水平方向上的缩放因子
android:textSize	setTextSize(float)	设置文本框内文本的字体大小
android:textStyle	setTypeface(Typeface)	设置文本框内文本的字体风格
android:typeface	setTypeface(Typeface)	设置文本框内文本的字体
android:width	setWidth(int)	设置文本框宽度,单位为 pixel

表 3-1 中 android:autoLink 属性值是一个或多个,如果是多个属性值则各值之间竖线隔开。

2. TextView 经典案例

下面通过几个简单的案件来演示利用 TextView 显示文字的技巧。

【例 3-1】 利用 TextView 控件,在屏幕上显示文字。

其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li3_1TextView1。

(2) 打开 res\layout 目录下的 main.xml 文件,在布局文件中声明 TextView 控件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = " # aabbcc">
<!-- TextView - 文本显示控件 -->
< TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:id = "@ + id/textView" />
    </LinearLayout >
```

(3) 打开 src\fs.li3_1textview1 包下的 MainActivity.java 文件,实现文字显示的具体功能,其代码为:

```
package fs.li3_1textview1;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends Activity {
    /** 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView tv = (TextView)this.findViewById(R.id.textView);
        //设置文本显示控件的文本内容,需要换行就用" "
        tv.setText("我是 TextView/n 显示文字用的");
    }
}
```

运行效果如图 3-1 所示。



图 3-1　使用 TextView 显示文字

【例 3-2】　本案例使用 TextView 在屏蔽上显示文字，并可以为其设置不同的背景色和字体颜色。

其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li3_2TextView2。

(2) 打开 res\layout 目录下的 main.xml 文件，用于声明两个 TextView 控件，其代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#000000">
    <TextView
        android:text = "@ + id/TextView1"
        android:id = "@ + id/TextView1"
        android:textSize = "36dip"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
    <TextView
        android:id = "@ + id/TextView2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/TextView1"
```

```
        android:layout_below = "@ + id/TextView1"
        android:layout_marginTop = "48dp"
        android:text = "@ + id/TextView2"
        android:textSize = "36dip" />
</RelativeLayout>
```

(3) 设置颜色文件 color. xml。打开 res/value 目录,选择 value 文件夹并右击,在弹出的快捷菜单中选择"新建"→"文件"选项,弹出"新建文件"窗口,在"文件名"文本框中输入对应的文件,如图 3-2 所示。



图 3-2 "新建文件"窗口

color. xml 文件的代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <color name = "red"># fd8d8d </color>
    <color name = "green"># 9cfda3 </color>
    <color name = "blue"># 8d9dfd </color>
    <color name = "white"># FFFFFF </color>
    <color name = "black"># 000000 </color>
</resources>
```

(4) 打开 src\fs. li3_2textview2 包下 MainActivity. java 文件,用于实现文字颜色设置的功能,其代码为:

```
package fs.li3_2textview2;
import android.app.Activity;
import android.content.res.Resources;
```

```
import android.graphics.Color;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends Activity {
    /** 第一次调用活动. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView tv01 = (TextView)this.findViewById(R.id.TextView1);
        tv01.setText("设置文字背景色");
        Resources resources = getBaseContext().getResources();
        //得到图片引用
        Drawable Hdrawable = resources.getDrawable(R.color.white);
        //设置图像为背景
        tv01.setBackgroundDrawable(Hdrawable);
        TextView tv02 = (TextView)this.findViewById(R.id.TextView2);
        tv02.setText("设置文字颜色");
        tv02.setTextColor(Color.RED);        //设置颜色
    }
}
```

运行效果如图 3-3 所示。



图 3-3　设置文字颜色效果

【例 3-3】　使用 TextView 控件，独特显示文字效果。

其具体实现步骤如下所示。

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li3_3TextView3。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明两个 TextView 控件，其代

码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#000000" >
    <TextView
        style = "@style/style1"
        android:text = "TextView 显示独特文字"
        android:textSize = "36dip"
        android:id = "@ + id/TextView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content">
    </TextView>
    <TextView
        android:id = "@ + id/TextView2"
        style = "@style/style2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentTop = "true"
        android:layout_centerHorizontal = "true"
        android:layout_marginTop = "114dp"
        android:text = "TextView 显示独特文字"
        android:textSize = "36dip" />
</RelativeLayout>
```

(3) 新建 style.xml 样式文件。在 res\value 目录下创建一个名为 style.xml 文件，用于实现样式的设置，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <style name = "style1">
        <item name = "android:textSize">16sp</item>
        <item name = "android:textColor">#FFFFFF</item>
    </style>
    <style name = "style2">
        <item name = "android:textSize">28sp</item>
        <item name = "android:textColor">#fd8d8d</item>
        <item name = "android:fromAlpha">0.5</item>
        <item name = "android:toAlpha">0.45</item>
    </style>
</resources>
```

(4) 打开 src\fs.li3_3textview3 包下的 MainActivity.java 文件，用于实现具体的功能，其代码为：

```
package fs.li3_3textview3;
import android.os.Bundle;
import android.app.Activity;
public class MainActivity extends Activity {
    @Override
```

```
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);   //跳转到主界面
        }
    }
```

运行效果如图 3-4 所示。



图 3-4　显示文字的独特效果

3.1.2　编辑框控件

1. EditeText 控件概述

EditeText 类继承自 TextView 类，EditText 类与 TextView 类最大的不同就是用户可以对 EditText 控件进行编辑。同时，用户还可以为 EditText 控件设置监听器，用来检测用户的输入是否合法等。表 3-2 列出了 EditText 类继承自 TextView 类中的常用属性及对应方法说明。

表 3-2　EditText 的 XML 属性及说明

XML 属性	相关方法	说明
android:cursorVisible	setcursorVisible (boolean)	设置光标是否可见，默认为可见
android:lines	setLines(int)	通过设置固定的行数来决定 EditText 的高度
android:maxLines	setMaxLines(int)	设置最大行数
android:minLines	setMinLines(int)	设置最小行数
android:password	setTransformationMethod (TransformationMethod)	设置文本框中的内容是否显示为密码
android:phoneNumber	setKeyListener (KeyListener)	设置文本框中的内容只能是电话号码

续表

XML 属性	相关方法	说明
android:scrollHorizontally	setHorizontallyScrolling (boolean)	设置文本框是否可以水平地进行滚动
android:selectAllOnFocus	setSelectAllOnFocus (boolean)	如果文本内容可选中,则当文本框获得焦点时自动选中全部文本内容
android:shadowColor	setShadowLayer(float, float,float,int)	为文本框设置指定颜色的阴影
android:shadowDx	setShadowLayer(float, float,float,int)	为文本框设置阴影的水平偏移,为浮点数
android:shadowDy	setShadowLayer(float, float,float,int)	为文本框设置阴影的垂直偏移,为浮点数
android:shadowRadius	setShadowLayer(float, float,float,int)	为文本框设置阴影的半径,为浮点数
android:singleLine	setTransformationMethod (TransformationMethod)	设置文本框为单行模式
android:maxLength	setFilters(InputFilter)	设置最大显示长度

2. EditText 经典案例

下面构建一个简单的登录界面演示 EditText 编辑框的用法。

【例 3-4】 简单的本地验证实例。

其具体操作步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li3_4EditText。

(2) 打开 res\layout 目录下的 main.xml 文件,在文件中定义三个线性布局,在布局文件中声明两个 EditText 控件、两个 TextView 控件和两个 Button 控件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
    <LinearLayout
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:background = "#ffcc66"
        android:paddingLeft = "5dip"
        android:paddingRight = "5dip"
        android:paddingTop = "5dip">
        <LinearLayout
            android:id = "@ + id/LinearLayout1"
            android:orientation = "horizontal"
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent">
            <TextView
                android:text = "用户名: "
                android:id = "@ + id/TextView2"
                android:textColor = "#333444"
```

```
            android:layout_width = "wrap_content"
            android:layout_height = "40dip"
            android:layout_marginLeft = "5dip"
            android:textSize = "18dip"
            android:gravity = "center_vertical"/>
        < EditText
            android:id = "@ + id/EditTextuid"
            android:singleLine = "true"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:layout_marginLeft = "0dip"
            android:text = "yonghuming"/>
    </LinearLayout >
    < LinearLayout
        android:id = "@ + id/LinearLayout2"
        android:orientation = "horizontal"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content">
        < TextView
            android:text = "密     码: "
            android:id = "@ + id/TextView3"
            android:textColor = "#222222"
            android:layout_width = "wrap_content"
            android:layout_height = "40dip"
            android:layout_marginLeft = "5dip"
            android:textSize = "18dip"
            android:gravity = "center_vertical"/>
        < EditText
            android:id = "@ + id/EditTextPwd"
            android:singleLine = "true"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
             android:text = "abcdef"/>
    </LinearLayout >
    < LinearLayout
        android:id = "@ + id/LinearLayout3"
        android:orientation = "horizontal"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content">
        < Button
            android:text = " 登录 "
            android:id = "@ + id/loginLog"
            android:layout_width = "75dip"
            android:layout_height = "40dip"
            android:textSize = "18dip"
            android:gravity = "center"/>
        < Button
            android:text = "清空 "
            android:id = "@ + id/loginClear"
            android:layout_width = "75dip"
            android:layout_height = "40dip"
            android:textSize = "18dip"
            android:gravity = "center"/>
    </LinearLayout >
  </LinearLayout >
</LinearLayout >
```

(3) 打开 src\fs. li3_4edittext 包下的 MainActivity. java 文件，在文件中实现 EditText 编辑框中输入内容，当用户名和密码正确时，单击“确定”按钮即弹出 Toast 提示“恭喜登录成功！”，否则将提示“请输入正确的用户名或密码！”。单击“清空”按钮，则会清空所填写的姓名和密码内容，其代码为：

```
package fs.li3_4edittext;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends Activity {
    /** 第一次调用活动. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button bLogin = (Button)this.findViewById(R.id.loginLog);          //登录按钮
        Button bClear = (Button)this.findViewById(R.id.loginClear);        //清空按钮
        final EditText eUid = (EditText)this.findViewById(R.id.EditTextuid); //用户名
        final EditText eMima = (EditText)this.findViewById(R.id.EditTextPwd);//密码
        bLogin.setOnClickListener                        //添加登录按钮监听器
        (
             new OnClickListener()
             {
                public void onClick(View v)
                {
                    String strUid = eUid.getText().toString().trim();
                    String strPwd = eMima.getText().toString().trim();
                    if(strUid.equals("yonghuming")&&strPwd.equals("abcdef"))
                    {
  Toast.makeText(MainActivity.this, "恭喜登录成功!", Toast.LENGTH_SHORT).show();
                    }
                    else
                    {
                Toast.makeText(MainActivity.this, "请输入正确的用户名或密码!", Toast.LENGTH
_SHORT).show();
                    }
                }
             });
        bClear.setOnClickListener                        //添加清空按钮监听器
        (
                new OnClickListener()
                {
                    public void onClick(View v) {
                        eUid.setText("");          //设置用户名为空
                        eMima.setText("");         //设置密码为空
                    }
                });
        }
}
```

运行效果如图 3-5 所示。



图 3-5　本地验证界面

3.1.3　自动提示文本框

1. AutoCompleteTextView 类概述

除了基本的文本控件外，在 Android 中还提供了自动提示文本框 AutoCompleteTextView。所谓的自动提示就是在文本框中输入文字时，会显示可能的关键字让用户来选择。在其他系统下完成此功能可能非常麻烦，但是在 Android 中是非常容易达到的。

AutoCompleteTextView 类继承 EditText 类，自动提示文本框的外观与图片文本框没有任何区别，只是当用户输入某些文字时，会自动出现下拉菜单，显示与用户输入文字相关的信息，用户直接单击需要的文字，即可自动填写到文本控件中。

AutoCompleteTextView 除了可使用 EditText 提供的 XML 属性和方法外，还支持如表 3-3 所示的常用 XML 属性及方法。

表 3-3　AutoCompleteTextView 支持的常用属性及方法

XML 属性	方　法	描　述
android:completionHint	setCompletionHint(CharSequence)	设置出现在下拉菜单中的提示标题
android:completionThreshold	setThreshold(int)	设置用户至少输入几个字符才会显示提示
android:dropDownHeight	setDropDownHeight(int)	设置下拉菜单的高度
android:dropDownHorizontalOffset		设置下拉菜单与文本框之间的水平偏移，下拉菜单默认与文本框左对齐
android:dropDownVerticalOffset		设置下拉菜单与文本框之间的垂直偏移，下拉菜单默认紧跟文本框
android:dropDownWidth	setDropWidth(int)	设置下拉菜单的宽度
android:popupBackground	setDropDownBackgroundResource(int)	设置下拉菜单的背景

使用 AutoCompleteTextView 很简单，只要为它设置一个适配器，该适配器封装了 AutoCompleteTextView 预设的提示文本。

2. AutoCompleteTextView 经典案例

下面通过一个简单的案例来演示 AutoCompleteTextView 的使用方法。

【例 3-5】 实现输入某个字符即出现对应的选择。

其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li3_5AutoCompleteTextView。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明一个 AutoCompleteTextView 控件，其代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc">
  <AutoCompleteTextView
      android:id = "@ + id/autoComplete"
      android:layout_width = "fill_parent"
      android:layout_height = "wrap_content"
      android:layout_alignLeft = "@ + id/textView1"
      android:layout_below = "@ + id/textView1"
      android:layout_marginTop = "27dp"
      android:ems = "10"/>
</RelativeLayout>
```

(3) 打开 src\fs.li3_5autocompletetextview 包下的 MainActivity.java 文件，在文件中实现文本的自动选择功能，其代码为：

```
package fs.li3_5autocompletetextview;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
//自动完成文本
public class MainActivity extends Activity {
    //字符选择
    private static final String[] COUNTRIES = {"China","Russia","Germany",
        "Ukraine","Belarus","USA","China1","China2","USA1"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //创建一个 ArrayAdapter
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_dropdown_item_1line,COUNTRIES);
        //获取 AutoCompleteTextView 对象
        AutoCompleteTextView autoComplete = (AutoCompleteTextView) findViewById(R.id.autoComplete);
        //将 AutoCompleteTextView 与 ArrayAdapter 进行绑定
```

```
            autoComplete.setAdapter(adapter);
            //设置 AutoCompleteTextView 输入一个字符就进行提示
            autoComplete.setThreshold(1);
        }
    }
```

运行程序，在文本框中输入某个字符，即显示相应的选择，效果如图 3-6 所示。



图 3-6　自动提示文本框

3.1.4　多行自动提示文本框

1. MultiAutoCompleteTextView 类概述

MultiAutoCompleteTextView 类是一个继承自 AutoCompleteTextView 的可编辑的文本视图，能够对用户输入的文本进行有效地扩充提示，而不需要用户输入整个内容（用户输入一部分内容，剩下的部分系统就会给予提示）。

用户必须提供一个 MultiAutoCompleteTextView.Tokenizer 用来区分不同的子串。

其重要方法主要有：

- enoughToFilter()：当文本长度超过阈值时过滤。
- performValidation()：代替验证整个文本，这个子类方法验证每个单独的文字标记。
- setTokenizer（MultiAutoCompleteTextView.Tokenizer t）：用户正在输入时，tokenizer 设置将用于确定文本相关范围内。

2. MultiAutoCompleteTextView 经典案例

下面通过一个案例来演示 MultiAutoCompleteTextView 类的用法。

【例 3-6】 使用 MultiAutoCompleteTextView 实现不同子串的选择。

其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li3_6MultiAutoCompleteTextView。

(2) 打开 res\layout 目录下的 main. xml 文件,在文件中声明一个 AutoCompleteTextView 控件和一个 MultiAutoCompleteTextView 控件,其代码为:

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc" >
<AutoCompleteTextView
     android:id = "@ + id/autoCompleteTextView"
     android:layout_width = "fill_parent"
     android:layout_height = "wrap_content"/>
<MultiAutoCompleteTextView
        android:id = "@ + id/multiAutoCompleteTextView"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"/>
</RelativeLayout>
```

(3) 打开 src\fs. li3_6multautocompletetextview 包下的 MainActivity. java 文件,实现区分不同子串的选择,其代码为:

```
package fs.li3_6multiautocompletetextview;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.MultiAutoCompleteTextView;
public class MainActivity extends Activity {
    //广东几个地市
    private static final String[] cities = new String[] {"ShenZen", "GuangZhou", "ChaoShan",
"HuiZhou","HeYuan", "MeiZhou", "QingYuan", "ShaoGuang","KaiPing"};
    private AutoCompleteTextView autoCompleteTextView = null;
    private MultiAutoCompleteTextView multiAutoCompleteTextView = null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    autoCompleteTextView = (AutoCompleteTextView)findViewById(R.id.autoCompleteTextView);
    multiAutoCompleteTextView = (MultiAutoCompleteTextView)
          findViewById(R.id.multiAutoCompleteTextView);
        //创建适配器
         ArrayAdapter < String > adapter = new ArrayAdapter < String >(this, android.R.layout.
simple_dropdown_item_1line, cities);
        autoCompleteTextView.setAdapter(adapter);
        //设置输入多少字符后提示,默认值为 2
        autoCompleteTextView.setThreshold(2);
        multiAutoCompleteTextView.setAdapter(adapter);
        multiAutoCompleteTextView.setThreshold(2);
    //用户必须提供一个 MultiAutoCompleteTextView.Tokenizer 用来区分不同的子串
        multiAutoCompleteTextView.setTokenizer(new MultiAutoCompleteTextView.CommaTokenizer());
    }
}
```

运行效果如图 3-7 所示。



图 3-7　多行自动提示文本框

3.2　按钮类控件

按钮类控件是窗口类名被系统预定义为 Button 的一类控件，该类控件具有十多种不同的窗口风格，包含了普通的下压式按钮、单选按钮、复选按钮和分组框等多种常用的按钮形式。

3.2.1　按钮控件

1. Button 控件概述

Button 控件继承 TextView 类，用户可以对 Button 控件进行按下或单击等操作，Button 控件的用法比较简单，主要是为 Button 控件设置 View. OnClickListener 监听器并在监听器的实现代码中开发按钮按下事件的处理代码。

Button 控件主要具有如下属性：

- Java 代码中通过 btn1 关联次控件 android:id="@+id/btn1"
- 控件宽度

```
android:layout_width = "80px"                  //"80dip"或"80dp"
android:layout_width = "wrap_content"
android:layout_width = "match_parent"
```

- 控件高度

```
android:layout_height = "80px"                 //"80dip"或"80dp"
android:layout_height = "wrap_content"
android:layout_height = "match_parent"
```

• 控件排布

```
android:orientation = "horizontal"
android:orientation = "vertical"
```

• 控件间距

```
android:layout_marginLeft = "5dip"            //距离左边
android:layout_marginRight = "5dip"           //距离右边
android:layout_marginTop = "5dip"             //距离上面
android:layout_marginRight = "5dip"           //距离下面
```

• 控件显示位置

```
android:gravity = "center"                    //左、右、上、下
android:gravity = "center_horizontal"
android:layout_gravity                        //是本元素对父元素的重力方向
android:layout_gravity                        //属性则设置控件本身相对于父控件的显示位置
android:gravity                               //是本元素所有子元素的重力方向
android:layout_gravity = "center_vertical"
android:layout_gravity = "left"
android:layout_gravity = "left|bottom"
```

• TextView 中文本字体

```
android:text = "@String/text1"                //在 string.xml 中定义 text1 的值
android:textSize = "20sp"
android:textColor = "#ff123456"
android:textStyle = "bold"                //普通(normal), 斜体(italic),粗斜体(bold_italic)
```

• 定义控件是否可见

```
android:visibility = "visible"                //可见
android:visibility = "invisible"              //不可见,但是在布局中占用的位置还在
android:visibility = "gone"                   //不可见,完全从布局中消失
```

• 定义背景图片

```
android:background = "@drawable/img_bg"   //img_bg 为 drawable 下的一张图片
```

• seekbar 控件背景图片及最大值

```
android:progressDrawable = "@drawable/seekbar_img"
android:thumb = "@drawable/thumb"
android:max = "60"
```

• 在父亲布局的相对位置

```
android:layout_alignParentLeft = "true"       //在布局左边
android:layout_alignParentRight = "true"      //在布局右边
android:layout_alignParentTop = "true"        //在布局上面
android:layout_alignParentBottom = "true "    //在布局的下面
```

• 在某个控件的相对位置

```
android:layout_toRightOf = "@id/button1"      //在控件 button1 的右边,不仅仅是紧靠着
android:layout_toLeftOf = "@id/button1"       //在控件 button2 的左边,不仅仅是紧靠着
android:layout_below = "@id/button1"          //在控件 button1 下面,不仅仅是正下方
android:layout_above = "@id/button1"          //在控件 button1 下面,不仅仅是正下方
```

• 定义和某控件对齐

```
android:layout_alignTop = "@id/button1"       //和控件 button1 上对齐
android:layout_alignBottom = "@id/button1"    //和控件 button1 下对齐
```

```
android:layout_alignLeft = "@id/button1"          //和控件 button1 左对齐
android:layout_alignRight = "@id/button1"         //和控件 button2 右对齐
android:layout_centerHorizontal = "true"          //水平居中
android:layout_centerVertical = "true"
android:layout_centerInParent = "true"
```

- 仅在 LinearLayout 中有效

```
android:layout_weight = "1"                       //设置控件在一排或一列中所占比例值
```

2. Button 控件经典案例

下面通过一个经典案例来演示 Button 控件的用法。

【例 3-7】 演示 Button 控件的用法,实现单击按钮时改变 Button 控件的背景。

其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li3_7Button。

(2) 打开 res\layout 目录下的 main. xml 文件,在文件中声明两个 Button 控件,其代码为:

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc" >
<Button
        android:id = "@ + id/button_first"
        android:layout_height = "wrap_content"
        android:layout_width = "wrap_content"
        android:text = "红色按钮"
        android:onClick = "changeButtonColor"/>
    <Button
        android:id = "@ + id/button_second"
        android:layout_height = "wrap_content"
        android:layout_width = "wrap_content"
        android:text = "蓝色按钮"
        android:layout_below = "@id/button_first"/>
</RelativeLayout>
```

(3) 在 res\value 目录下创建一个 color. xml 文件,用于设置按钮背景颜色,其代码为:

```
<?xml version = "1.0" encoding = "UTF - 8"?>
<resources>
    <color name = "red">#f00</color>
    <color name = "green">#0f0</color>
    <color name = "blue">#00f</color>
    <color name = "black">#000</color>
</resources>
```

(4) 打开 src\fs. li3_7button 包下的 MainActivity. java 文件,用于实现单击按钮改变按钮的背景色,其代码为:

```
package fs.li3_7button;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class MainActivity extends Activity
{
    private Button button01 = null;
    private Button button02 = null;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        button01 = (Button)findViewById(R.id.button_first);
        button02 = (Button)findViewById(R.id.button_second);
        //绑定事件源和监听器对象
        button02.setOnClickListener(new MyButtonListener());
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
    //按钮 1 的点击事件
    public void changeButtonColor(View view)
    {
        button01.setBackgroundColor(getResources().getColor(R.color.red));
    }
    //内部类,实现 OnClickListener 接口
    //作为第二个按钮的监听器类
    class MyButtonListener implements OnClickListener
    {
        public void onClick(View v)
        {
            button02.setBackgroundColor(getResources().getColor(R.color.blue));
        }
    }
}
```

运行效果如图 3-8(a)所示,单击按钮时,即可改变按钮的背景颜色,如图 3-8(b)所示。

3.2.2 图片按钮控件

1. ImageButton 控件概述

ImageButton 控件继承自 ImageView 类,ImageButton 控件与 Button 控件的主要区别在 ImageButton 中没有 text 属性,即按钮将显示的图片而不是文本。在 ImageButton 中设置按钮显示的图片可以通过 android:src 属性来设置,也可通过 setImageResource(int)方法来设置。

默认情况下,ImageButton 同 Button 一样具有背景色,当按钮处于不同的状态(如按下



(a) 默认效果



(b) 改变按钮背景色

图 3-8　按钮用法

等)时,背景色也会随之变化。当 ImageButton 所显示的图片不能完全覆盖背景色时,这种显示效果将会非常糟糕,所以使用 ImageButton 一般要将背景色设置为其他图片或直接设置为透明的。

不管怎样,都需要为按钮控件指定不同状态下显示的图片,否则用户将无法区别是否按下了按钮。设置按钮在不同状态下显示不同的图片可以通过编写 XML 文件来实现。

2. ImageButton 控件经典案例

下面通过一个实例来演示 ImageButton 控件的使用。

【例 3-8】 使用 ImageButton 控件设置按钮的背景图片。

其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li3_8ImageButton。

(2) 打开 res\layout 目录下的 main. xml 文件,在文件中布局 4 个 ImageButton 控件,其代码为:

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#000000">
<ImageButton
    android:id = "@+id/ImageButton4"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_alignLeft = "@+id/ImageButton2"
```

```
        android:layout_alignParentBottom = "true"
        android:layout_marginBottom = "14dp"
        android:src = "@drawable/bt4" />
    < ImageButton
        android:id = "@ + id/ImageButton1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_above = "@ + id/ImageButton2"
        android:layout_alignLeft = "@ + id/ImageButton2"
        android:layout_marginBottom = "20dp"
        android:src = "@drawable/bt1" />
    < ImageButton
        android:id = "@ + id/ImageButton2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_above = "@ + id/ImageButton3"
        android:layout_alignParentLeft = "true"
        android:layout_marginBottom = "18dp"
        android:layout_marginLeft = "14dp"
        android:src = "@drawable/bt2" />
    < ImageButton
        android:id = "@ + id/ImageButton3"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_above = "@ + id/ImageButton4"
        android:layout_alignLeft = "@ + id/ImageButton2"
        android:layout_marginBottom = "19dp"/>
    </RelativeLayout >
```

(3) 打开 res\drawable-mdpi 文件,在文件中放置三个图片,用于设置图片按钮。

(4) 打开 src\fs.li3_8imagebutton 包下的 MainActivity.java 文件,在文件中实现单击对应的图片按钮,弹出对应的对话框,其代码为:

```
package fs.li3_8imagebutton;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.AlertDialog.Builder;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.TextView;
public class MainActivity extends Activity {
    /** 第一次调用活动. */
    TextView textView;
    ImageButton imageButton1,imageButton2,imageButton3,imageButton4;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        imageButton1 = (ImageButton)findViewById(R.id.ImageButton1);
        imageButton2 = (ImageButton)findViewById(R.id.ImageButton2);
        imageButton3 = (ImageButton)findViewById(R.id.ImageButton3);
        imageButton4 = (ImageButton)findViewById(R.id.ImageButton4);
```

```
        //给按钮设置使用的图标,由于 button1,button2,button3,已经在 xml 文件中设置这里就不
        //设置了
        imageButton4.setImageDrawable(getResources().getDrawable(android.R.drawable.sym_call
_incoming));
        //以下分别为每个按钮设置事件监听 setOnClickListener
        imageButton1.setOnClickListener(new Button.OnClickListener(){
            public void onClick(View v) {
                //对话框,Builder 是 AlertDialog 的静态内部类
                Dialog dialog = new AlertDialog.Builder(MainActivity.this)
                //设置对话框的标题
                    .setTitle("小航提示")
                    //设置对话框要显示的消息
                    .setMessage("我是 ImageButton4")
                    //给对话框加个按钮叫"确定",并且设置监听器
             .setPositiveButton("确定", new DialogInterface.OnClickListener(){
                        public void onClick(DialogInterface dialog, int which) {
                            //单击"确定"按钮之后要执行的操作就写在这里
                        }
                    }).create();                //创建按钮
                    dialog.show();              //显示
            }
        });
        imageButton2.setOnClickListener(new Button.OnClickListener(){
            public void onClick(View v) {
                Builder dialog = new AlertDialog.Builder(MainActivity.this);
                dialog.setTitle("提示");
             dialog.setMessage("我是 ImageButton2,我要使用 ImageButton3 的图标");
             dialog.setPositiveButton("确定", new DialogInterface.OnClickListener(){
                    public void onClick(DialogInterface dialog, int which) {
                     //成功把 Button3 的图标掠夺过来
             imageButton2.setImageDrawable(getResources().getDrawable(R.drawable.bt3));
                 }
                }).create();                    //创建按钮
                dialog.show();
            }
        });
        imageButton3.setOnClickListener(new Button.OnClickListener(){
            public void onClick(View v) {
                Builder dialog = new AlertDialog.Builder(MainActivity.this);
                dialog.setTitle("提示");
                dialog.setMessage("我是 ImageButton1");
                dialog.setPositiveButton("确定", new DialogInterface.OnClickListener(){
                    public void onClick(DialogInterface dialog, int which) {
                        //把 imageButton3 的图标设置为系统的打电话图标
        imageButton3.setImageDrawable(getResources().getDrawable(android.R.drawable.sym_
action_call));
                    }
                }).create();                    //创建按钮
                dialog.show();
            }
            });
        imageButton4.setOnClickListener(new Button.OnClickListener(){
            public void onClick(View v) {
                Builder dialog = new AlertDialog.Builder(MainActivity.this);
                dialog.setTitle("提示");
```

```
                dialog.setMessage("使用的是系统图标");
                dialog.setPositiveButton("确定", new DialogInterface.OnClickListener(){
                    public void onClick(DialogInterface dialog, int which) {
                    }
                }).create();                    //创建按钮
                dialog.show();
            }
        });
    }
}
```

运行程序，效果如图 3-9(a)所示，单击第一个图标按钮时，效果如图 3-9(b)所示，单击第三个按钮的效果如图 3-9(c)所示。



(a) 默认效果　　(b) 图标按钮1　　(c) 图标按钮3

图 3-9　图标按钮的使用

3.2.3　双按钮控件

1. ToggleButton 控件概述

ToggleButton(双按钮)的继承关系如图 3-10 所示。ToggleButton 的状态只能是选中和未选中状态，并且需要为不同的状态设置不同的显示文本。除了继承自父类的一些属性和方法外，ToggleButton 也具有一些自己的 ToggleButton 属性，如表 3-4 所示。

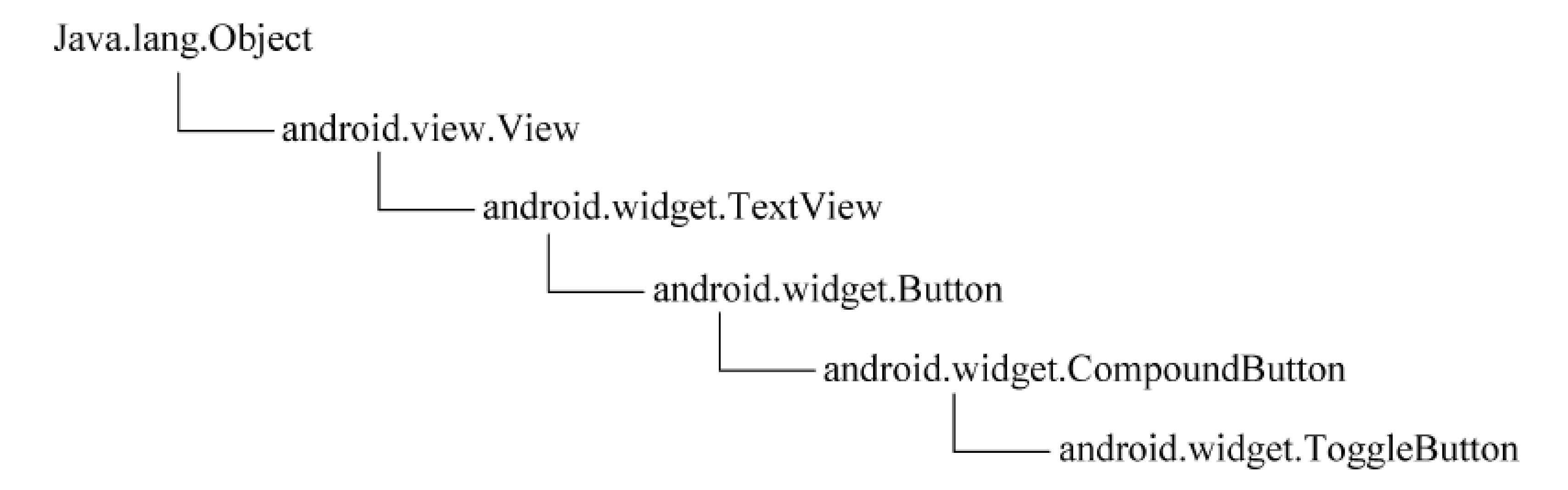


图 3-10　双按钮的继承关系

表 3-4　ToggleButton 支持的 XML 属性及描述

XML 属性	方　法	描　述
android:checked	setChecked(Boolean)	设置该按钮是否被选中
android:textOff		设置当该按钮没有被选中时显示的文本
android:textOn		设置当该按钮被选中时显示的文本

2. ToggleButton 控件经典案例

下面通过一个实例来演示 ToggleButton 控件的用法。

【例 3-9】 利用 ToggleButton 控件实现灯泡的开与关。

其具体操作步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li3_9ToggleButton。

(2) 打开 res\layout 目录下的 main.xml 文件,在文件中声明一个 ToggleButton 控件及一个 ImageView 控件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<!-- 声明一个垂直分布的线性分布 -->
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
android:orientation = "vertical"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent"
android:background = "@drawable/bj1">
<!-- 声明一个 ImageView 控件,该控制将会根据 ToggleButton 的状态显示不同的图片 -->
<ImageView
    android:id = "@ + id/imageView"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:src = "@drawable/bulb_off"
    android:layout_gravity = "center_horizontal"/>
<!-- 声明一个 ToggleButton 控件,设置了该控件在选中和未选中状态下显示的字符串 -->
<ToggleButton
    android:id = "@ + id/toggleButton"
    android:layout_width = "140dip"
    android:layout_height = "wrap_content"
     android:textOn = "开灯"
     android:textOff = "关灯"
     android:layout_gravity = "center_horizontal"/>
</LinearLayout>
```

(3) 打开 src/fs.li3_9toggle 目录下的 MainActivity.java,在文件中实现利用双按钮控制灯的开与关,其代码为:

```
import android.app.Activity;
import android.os.Bundle;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.ImageView;
import android.widget.ToggleButton;
public class MainActivity extends Activity {
    private ImageView imageView = null;
    private ToggleButton toggleButton = null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            //设置图铃状态
            imageView = (ImageView) findViewById(R.id.imageView);
            //设置 ToggleButton 状态
            toggleButton = (ToggleButton)findViewById(R.id.toggleButton);
            toggleButton.setOnCheckedChangeListener(new OnCheckedChangeListener(){
                //重写 onCheckedChanged 方法
                public void onCheckedChanged(CompoundButton buttonView, boolean isChecked){
                toggleButton.setChecked(isChecked); //控制控件状态
                //设置图像资源
                imageView.setImageResource(isChecked?R.drawable.bulb_off:R.drawable.bulb_on);
                }
            });
        }
    }
```

运行程序,效果如图 3-11(a)所示,单击图中的"开灯"按钮,效果如图 3-11(b)所示。



(a) 默认状态



(b) 开灯效果

图 3-11 双按钮控件

3.2.4 单复选按钮控件

1. RadioButton 与 CheckBox 控件概述

CheckBox(复选按钮)类和 RadioButton(单选按钮)类的继承关系如图 3-12 所示。CheckBox 控件和 RadioButton 控件都只有选中和未选中两种状态,不同的是 RadioButton 是单选按钮,它需要编制一个 RadioButton 中,同一时刻一个 RadioGroup 中只能有一个按钮处于选中状态。

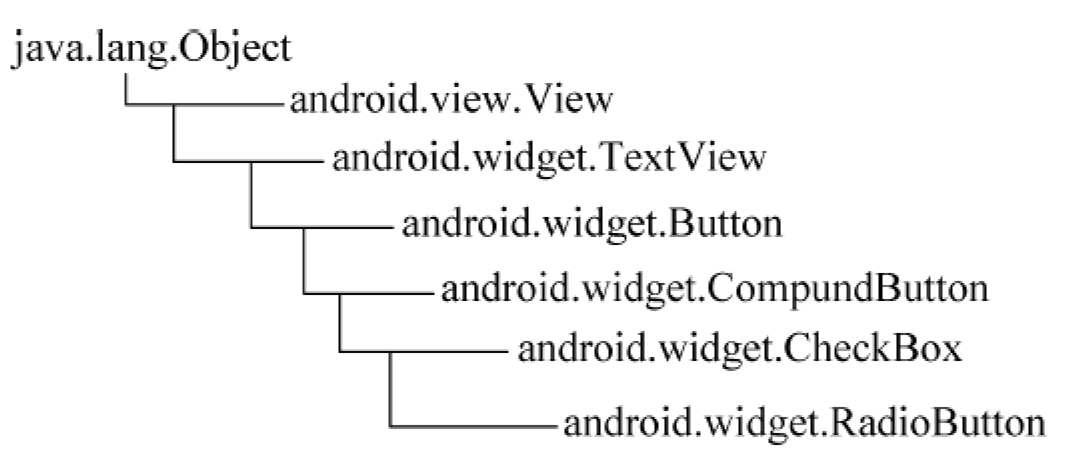


图 3-12 CheckBox 和 RadioButton 类的继承关系

RadioButton 与 CheckBox 都从父类 CompundButton 中继承了一些成员方法,这

些成员方法及说明如表 3-5 所示。

表 3-5　RadioButton 与 CheckBox 常用方法及说明

名　　称	说　　明
isCheck()	判断是否被选中，如果被选中返回 true，否则返回 false
performClick()	调用 OnClickListener 监听器，即模拟一次单击
setChecked(boolean checked)	通过传入的参数设置控件状态
toggle()	设置反控件当前的状态
setOnCheckedChangleListener(CompoundButton.OnCheckedChangleListener listener)	为控件设置 OnCheckedChangeListener 监听器

2．RadioButton 与 CheckBox 控件经典案例

下面通过两个案例来演示 RadioButton 与 CheckBox 控件的用法。

【例 3-10】 使用 RadioButton 控件实现颜色的选择。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li3_10 RadioButton。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明一个 RadioGroup 组件，在该控件中声明两个 RadioButton 控件，其代码为：

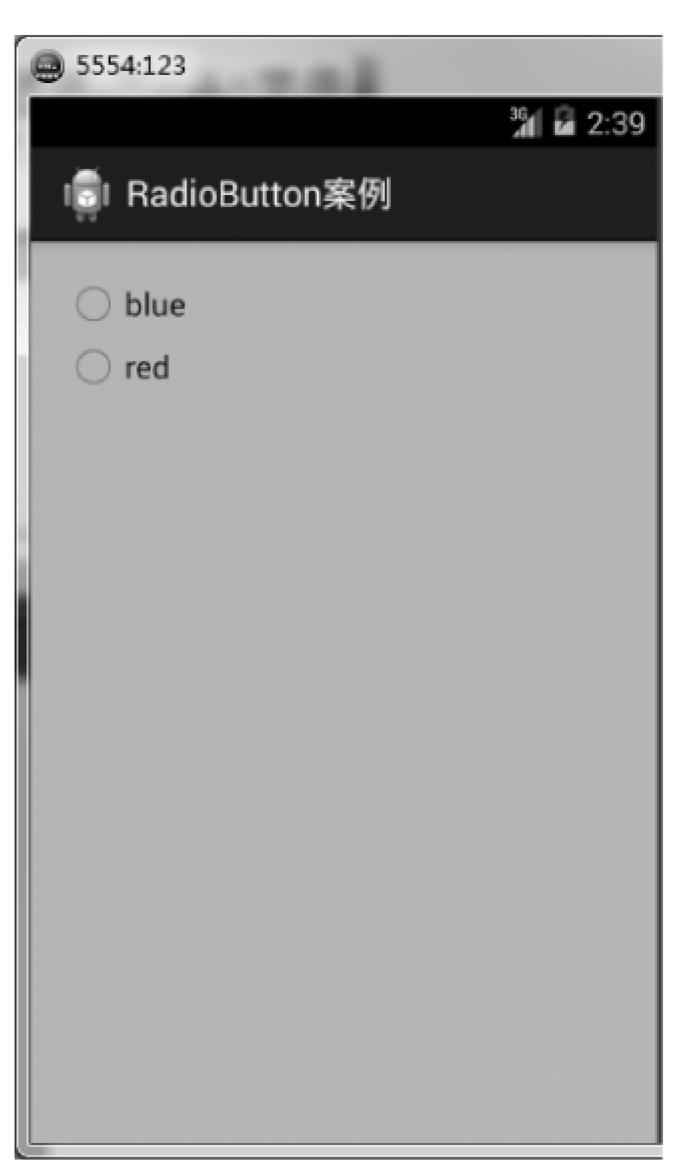
```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc" >
  <RadioGroup
    android:id = "@ + id/radioGroup"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:orientation = "vertical">
<RadioButton
    android:id = "@ + id/radioBlue"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "blue"/>
<RadioButton
    android:id = "@ + id/radioRed"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "red"/>
</RadioGroup>
</RelativeLayout>
```

(3) 打开 src\fs.li3_10radiogroup 目录下的 MainActivity.java 文件，在文件中当单击其中一个单选按钮时，即弹出对应的 Toast 提示信息，其代码为：

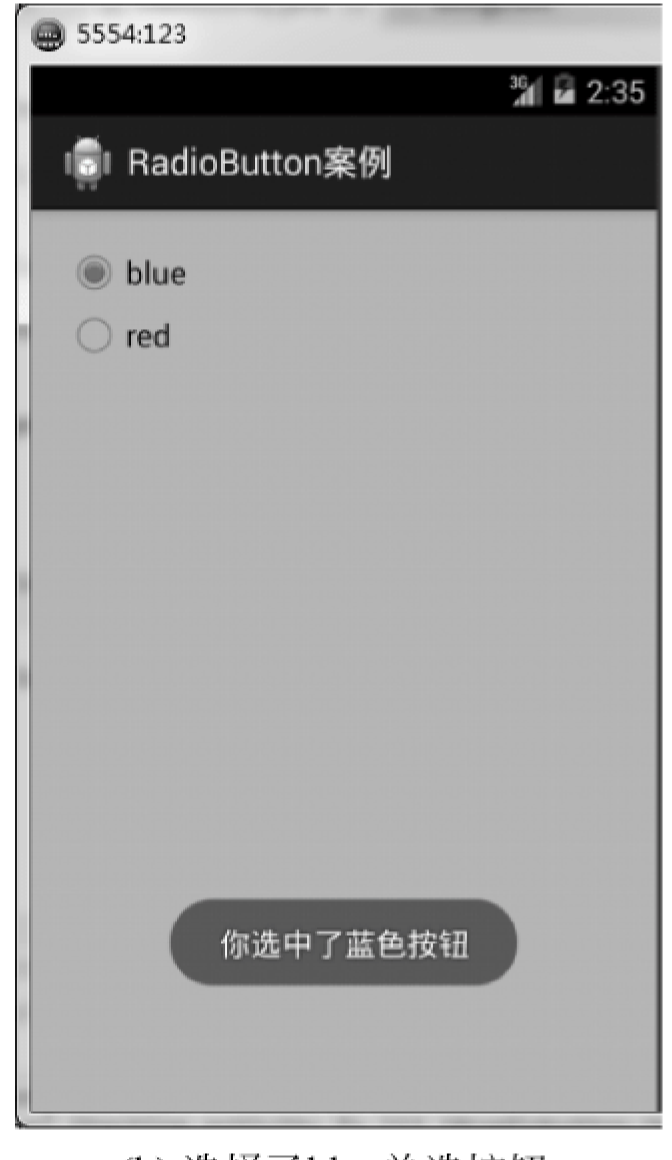
```
package fs.li3_10radiobutton;
import android.app.Activity;
```

```
import android.os.Bundle;
import android.widget.RadioGroup;
import android.widget.Toast;
public class MainActivity extends Activity
{
    /** 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
         final RadioGroup group = (RadioGroup)findViewById(R.id.radioGroup);
         group.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener()
         {
            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId)
            {
        switch(checkedId)
        {
        case R.id.radioBlue:
            Toast.makeText(getApplicationContext(), "你选中了蓝色按钮", Toast.LENGTH_LONG)
.show();
            break;
        case R.id.radioRed:
            Toast.makeText(getApplicationContext(), "你选中了红色按钮", Toast.LENGTH_LONG)
.show();
            break;
        }
            }
        });
    }
}
```

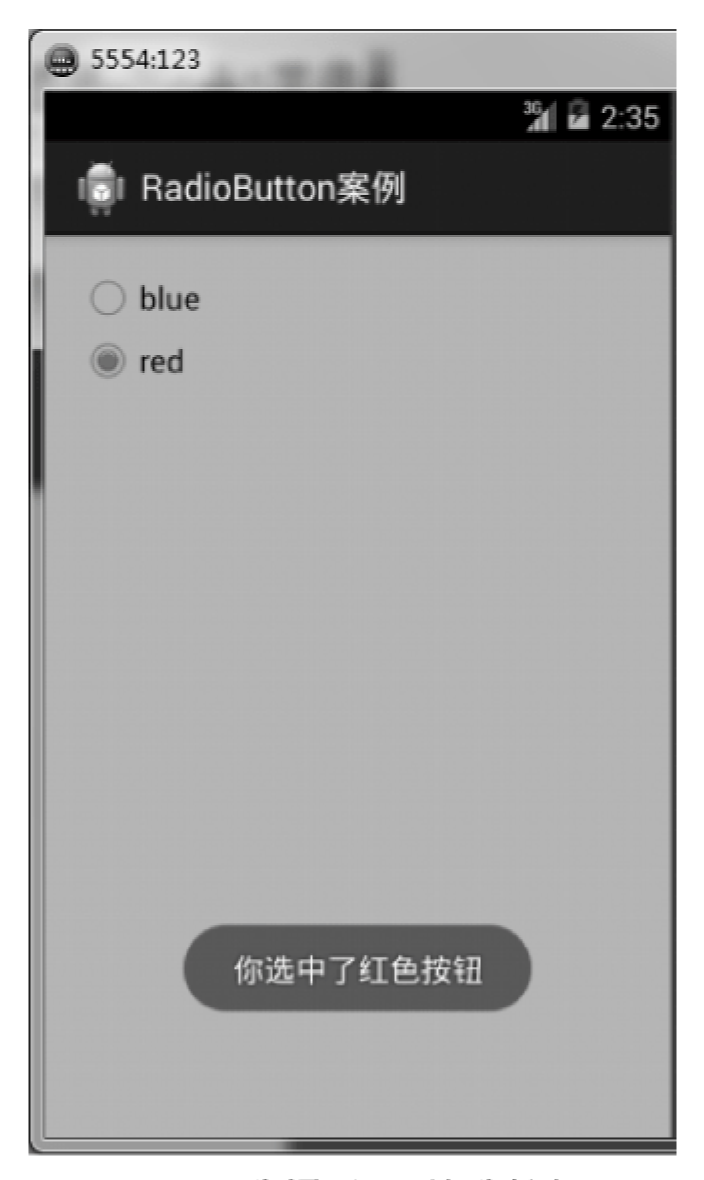
运行程序，默认效果如图 3-13 所示。



(a) 默认界面



(b) 选择了blue单选按钮



(c) 选择了red单选按钮

图 3-13　单选按钮使用

【例 3-11】 使用 CheckBox 控件实现多个选项的选择。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li3_11CheckBox。

(2) 打开 res\layout 目录下的 main.xml 文件，为线性布局，在文件中声明一个 TextView 控件、5 个 CheckBox 控件和一个 Button 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc" >
< TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "你知道的智能手机系统"/>
< CheckBox
    android:id = "@ + id/check1"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "苹果 IOS"   />
< CheckBox
    android:id = "@ + id/check2"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "谷歌 Android"   />
< CheckBox
    android:id = "@ + id/check3"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "RIM BlackBerry"   />
< CheckBox
    android:id = "@ + id/check4"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "微软 Windows Phone 7"   />
< CheckBox
    android:id = "@ + id/check5"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "诺基亚 Symbian"   />
< Button
    android:id = "@ + id/mybutton"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "确定"   />
</LinearLayout >
```

(3) 打开 src\fs.li3_11checkbox 包下的 MainActivity.java 文件，在文件中实现多项的选择，并每选择一个多选项时弹出相应的 Toast 提示信息，其代码为：

```
package fs.li3_11checkbox;
import android.app.Activity;
import android.os.Bundle;
import android.widget.CheckBox;
import android.widget.CompoundButton;
```

```
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.Toast;
public class MainActivity extends Activity
{
    /** 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    final CheckBox check1 = (CheckBox)findViewById(R.id.check1);
    final CheckBox check2 = (CheckBox)findViewById(R.id.check2);
    final CheckBox check3 = (CheckBox)findViewById(R.id.check3);
    final CheckBox check4 = (CheckBox)findViewById(R.id.check4);
    final CheckBox check5 = (CheckBox)findViewById(R.id.check5);
        //创建 CheckBox 事件监听器
         check1.setOnCheckedChangeListener(listener);
         check2.setOnCheckedChangeListener(listener);
         check3.setOnCheckedChangeListener(listener);
         check4.setOnCheckedChangeListener(listener);
         check5.setOnCheckedChangeListener(listener);
    }
    private OnCheckedChangeListener listener = new OnCheckedChangeListener()
    {
    @Override
    public void onCheckedChanged(CompoundButton buttonView,boolean isChecked)
    {
        switch(buttonView.getId())
        {
        case R.id.check1:
            if(isChecked)
                 Toast.makeText(getApplicationContext(), "你知道苹果 IOS 智能手机系统",
Toast.LENGTH_LONG).show();
            break;
        case R.id.check2:
            if(isChecked)
                Toast.makeText(getApplicationContext(), "你知道谷歌 Android 智能手机系统",
Toast.LENGTH_LONG).show();
            break;
        case R.id.check3:
            if(isChecked)
                 Toast.makeText(getApplicationContext(), "你知道 RIM BlackBerry 智能手机系
统",Toast.LENGTH_LONG).show();
            break;
        case R.id.check4:
            if(isChecked)
                Toast.makeText(getApplicationContext(), "你知道微软 Windows phone 7 智能手机
系统", Toast.LENGTH_LONG).show();
            break;
        case R.id.check5:
            if(isChecked)
                Toast.makeText(getApplicationContext(), "你知道诺基亚 Symbian 智能手机系统",
Toast.LENGTH_LONG).show();
            break;
            }
```

```
        }
    };
}
```

运行程序,效果如图 3-14 所示。



图 3-14　多选按钮的使用

3.3　时钟类控件

时钟类控件是 Android 用户界面中比较简单的控件,时钟控件包括 AnalogClock 控件和 DigitalClock 控件。

3.3.1　模拟时钟和数字时钟控件

1. AnalogClock 和 DigitalClock 控件概述

AnalogClock 视图显示了一个模拟的时钟,其中有一个时针和一个分针。与其相对的是 DigitalClock 视图,它可以显示数字模拟时钟,可精确到秒。这两个视图只能显示系统时间,不允许显示一个特定时区的时间。因此,如果想要显示一个特定时区的时间,那么就得去实现用户的自定义控件了。

AnalogClock 和 DigitalClock 的继承关系如图 3-15 所示。

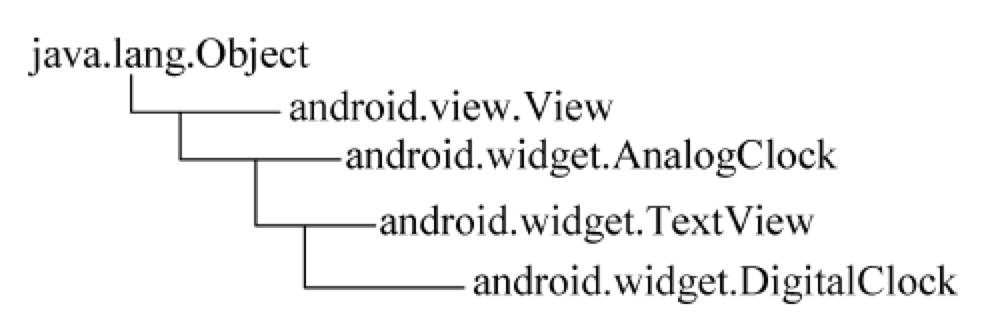


图 3-15　AnalogClock 和 DigitalClock 类的继承关系

它们在具体实现上,使用到如下三个对象。

(1) android. os. Handler: 通过产生的 Thread 对象在进程内同步调用方法 System .currentTimeMillis(),这样可以取得系统时间。

(2) java. lang. Thread: 为联系 Activity 与

Thread 的桥梁。

(3) android. os. Message：使用 Message 对象通知 Handler 对象，在收到 Message 对象后将时间变量的值显示在 TextView 中，这样即实现了数字时钟功能。

2. AnalogClock 和 DigitalClock 控件经典案例

下面通过两个案例分别来演示 AnalogClock 和 DigitalClock 控件的用法。

【例 3-12】 利用 AnalogClock 控件实现数字时钟及模拟时钟。其实现操作步骤为：

(1) 在 Eclipse 环境下创建一个 Android 应用项目，命名为 li3_12shizhong 的工程名。

(2) 编写布局文件，布局一个文本框及一个模拟时钟控件。打开 res/layout 目录下的 main. xml 文件，其代码修改为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout
  android:id = "@ + id/widget27"
  android:layout_width = "fill_parent"
  android:layout_height = "fill_parent"
  xmlns:android = "http://schemas.android.com/apk/res/android"
  android:orientation = "vertical"
  android:background = "@drawable/fr">
< AnalogClock
  android:id = "@ + id/myAnalogClock"
  android:layout_width = "wrap_content"
  android:layout_height = "wrap_content"
  android:layout_gravity = "center_horizontal">
</AnalogClock >
< TextView
  android:id = "@ + id/myTextView"
  android:layout_width = "wrap_content"
  android:layout_height = "wrap_content"
  android:text = "TextView"
  android:textSize = "20sp"
  android:textColor = "@drawable/white"
  android:layout_gravity = "center_horizontal">
</TextView >
</LinearLayout >
```

(3) 选择 res 目录下的 value 并右击，在弹出的快捷菜单中选择“新建”→“文件”选项，在“新建文件”界面左下侧的“文件名”文本框中输入 color. xml，即可完成 color. xml 文件的新建，其代码修改为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
   < drawable name = "white"># FF1F0F </drawable >
</resources >
```

(4) 编写 Activity 文件，实现时钟控件。打开 src/fs. li3_12shizhong 包下的 MainActivity. java 文件，代码修改为：

```
package fs.li3_12shizhong;
import android.app.Activity;
import android.os.Bundle;
/* 这里需要使用 Handler 类与 Message 类来处理运行线程 */
import android.os.Handler;
import android.os.Message;
```

```
import android.widget.AnalogClock;
import android.widget.TextView;
/*需要使用Java的Calendar与Thread类来取得系统时间*/
import java.util.Calendar;
import java.lang.Thread;
public class MainActivity extends Activity
{
  /*声明一常数作为判别信息用*/
  protected static final int GUINOTIFIER = 0x1234;
  /*声明两个widget对象变量*/
  private TextView mTextView;
  public AnalogClock mAnalogClock;
  /*声明与时间相关的变量*/
  public Calendar mCalendar;
  public int mMinutes;
  public int mHour;
  /*声明关键Handler与Thread变量*/
  public Handler mHandler;
  private Thread mClockThread;
  /** 第一次调用活动 */
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    /*通过findViewById取得两个widget对象*/
    mTextView = (TextView)findViewById(R.id.myTextView);
    mAnalogClock = (AnalogClock)findViewById(R.id.myAnalogClock);
    /*通过Handler来接收运行线程所传递的信息并更新TextView*/
    mHandler = new Handler()
    {
      public void handleMessage(Message msg)
      {
        /*这里是处理信息的方法*/
        switch (msg.what)
        {
          case MainActivity.GUINOTIFIER:
          /* 处理要TextView对象Show时间的事件 */
            mTextView.setText(mHour + " : " + mMinutes);
            break;
        }
        super.handleMessage(msg);
      }
    };
    /*通过运行线程来持续取得系统时间*/
     mClockThread = new LooperThread();
     mClockThread.start();
  }
  /*改写一个Thread Class用来持续取得系统时间*/
  class LooperThread extends Thread
  {
    public void run()
    {
      super.run();
      try
      {
```

```
            do
            {
              /* 取得系统时间 */
              long time = System.currentTimeMillis();
              /* 通过 Calendar 对象来取得小时与分钟 */
              final Calendar mCalendar = Calendar.getInstance();
              mCalendar.setTimeInMillis(time);
              mHour = mCalendar.get(Calendar.HOUR);
              mMinutes = mCalendar.get(Calendar.MINUTE);
              /* 让运行线程休息一秒 */
              Thread.sleep(1000);
               /* 重要关键程序:取得时间后发出信息给 Handler */
               Message m = new Message();
               m.what = MainActivity.GUINOTIFIER;
               MainActivity.this.mHandler.sendMessage(m);
            }while(MainActivity.LooperThread.interrupted() == false);
            /* 当系统发出中断信息时停止本循环 */
          }
    catch(Exception e)
          {
            e.printStackTrace();
          }
        }
      }
    }
```

运行程序，效果如图 3-16 所示。



图 3-16　数字时钟效果图

【例 3-13】 利用 DigitalClock 控件，设置像手机主界面以字符串显示的时间样式的一个数字时钟。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li3_13DigitalClock。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明一个模拟时钟及一个数字时钟，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:gravity = "center_horizontal"
    android:background = "#aabbcc" >
<!-- 定义模拟时钟 -->
<AnalogClock
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content" />
<!-- 定义数字时钟 -->
<DigitalClock
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
```

```
    android:textSize = "14pt"/>
</LinearLayout >
```

其他代码采用默认值,运行程序,效果如图 3-17 所示。



图 3-17　数字时钟设置

3.3.2　日期与时间控件

本小节将介绍日期与时间选择控件,首先对 DatePicker 类和 TimerPicker 类进行简单介绍,然后通过两个案例来说明怎样在程序中使用日期和时间选择控件。

1. DatePicker 控件概述

DatePicker 类继承自 FrameLayout 类,日期选择控件主要的功能向用户提供包含了年月日的日期数据并允许用户对其进行选择。如果要捕获用户修改日期选择控件中数据的事件,需要为 DatePicker 添加 onDateChangedListener 监听器。DatePicker 类的主要成员方法如表 3-6 所示。

表 3-6　DatePicker 类主要的成员方法及说明

名　称	说　明
getDayOfMonth()	获取日期天数
getMonth()	获取日期月份
getYear()	获取日期年份
init(int year,int monthOfYear,int dayOfMonth,DatePicker.OnDateChangedListener onDateChangedListener)	初始化 DatePicker 控件的属性,参数 onDateChangedListener 为监听器对象,负责监听日期数据的变化
setEnabled(boolean enabled)	根据传入的参数设置日期选择控件是否可用
updateDate(int year,int monthOfYear,int dayOfMonth)	根据传入的参数更新日期选择控件的各个属性值

2. TimePicker 类概述

TimePicker 同样继承 FrameLayout 类，时间选择控件向用户显示一天中的时间（可以为24小时制，也可以为 AM/PM 制），并允许用户进行选择。如果要捕获用户修改时间数据的事件，即需要为 TimePicker 添加 OnTimeChangedListener 监听器。TimePicker 类的主要成员方法如表 3-7 所示。

表 3-7 TimePicker 类主要的成员方法及说明

名 称	说 明
getCurrentHour()	获取时间选择控件的当前小时，返回 Integer 对象
getCurrentMinute()	获取时间选择控件的当前分钟，返回 Integer 对象
is24HourView()	判断时间选择控件是否为 24 小时制
setCurrentHour(Integer currentHour)	设置时间选择控件的当前小时，传入 Integer 对象
setCurrentMinute(Integer currentMinute)	设置时间选择控件的当前分钟，传入 Integer 对象
setEnabled(boolean enabled)	根据传入的参数设置时间选择控件是否可用
setIs24HourView(boolean is24HourView)	将时间设置为 24 小时制
setOnTimeChangedListener(TimePicker.OnTimeChangedListener onTimeChangedListener)	为时间选择控件添加 OnTimeChangedListener 监听器

3. DatePicker 和 TimerPicker 控件经典案例

下面通过两个案例来演示 DatePicker 和 TimerPicker 控件的用法。

【例 3-14】 本实例使用 DatePicker 设置日期。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li3_14DatePicker。

(2) 打开 res\layout 目录下的 main. xml 文件，在文件中声明一个 DatePicker 控件及一个 Button 控件，其代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc" >
<DatePicker
        android:id = "@ + id/datePick1"
        android:layout_height = "wrap_content"
        android:layout_width = "match_parent" />
    <Button
        android:id = "@ + id/button1"
        android:layout_below = "@id/datePick1"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:text = "获取 DatePicker 的值"/>
</RelativeLayout>
```

(3) 打开 src\fs. li3_14datepicker 包下的 MainActivity. java 文件，在文件中实现日期的设置，其代码为：

```
package fs.li3_14datepicker;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.DatePicker;
public class MainActivity extends Activity {
    private DatePicker datePicker1;
    private Button button1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        datePicker1 = (DatePicker)findViewById(R.id.datePick1);
        //设置默认的时间,如 2055 年 9 月 9 日
        datePicker1.updateDate(2012, 8, 9);
        button1 = (Button)findViewById(R.id.button1);
        OnClicLisers   cl = new OnClicLisers();
        button1.setOnClickListener(cl);
    }
    class OnClicLisers implements OnClickListener{
        @Override
        public void onClick(View v) {
            //TODO 自动存根法
            int y = datePicker1.getYear();
            int m = datePicker1.getMonth() + 1;
            int d = datePicker1.getDayOfMonth();
            System.out.println("y:" + y + " m:" + m + " d:" + d);
        }
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

运行程序,效果如图 3-18 所示。



图 3-18　日期的设置

【例 3-15】　本实例使用 TimePicker 设置时间。其实现的操作步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li3_15TimePicker。

(2) 打开 res\layout 目录下的 main.xml 文件,在文件中声明一个 TimePicker 控件及一个 Button 控件,其代码为:

```
<RelativeLayout xmlns:android = "http://schemas.
android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
```

```
        android:paddingBottom = "@dimen/activity_vertical_margin"
        android:paddingLeft = "@dimen/activity_horizontal_margin"
        android:paddingRight = "@dimen/activity_horizontal_margin"
        android:paddingTop = "@dimen/activity_vertical_margin"
        tools:context = ".MainActivity" >
        < TimePicker
            android:id = "@ + id/timePic1"
            android:layout_height = "wrap_content"
            android:layout_width = "match_parent"/>
        < Button
            android:id = "@ + id/buttone1"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:layout_below = "@id/timePic1"
            android:text = "获取 TimePick 时间"/>
    </RelativeLayout >
```

(3) 打开 src\fs. li3_15timerpicker 包下的 MainActivity. java 文件，在文件中实现时间的设置，其代码为：

```
package fs.li3_15timepicker;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TimePicker;
import android.widget.TimePicker.OnTimeChangedListener;
public class MainActivity extends Activity {
    private TimePicker timePick1;
    private Button buttone1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        timePick1 = (TimePicker)findViewById(R.id.timePic1);
        buttone1 = (Button)findViewById(R.id.buttone1);
        OnChangeListener  buc = new OnChangeListener();
        buttone1.setOnClickListener(buc);
        //是否使用 24 小时制
        timePick1.setIs24HourView(true);
        TimeListener times = new TimeListener();
        timePick1.setOnTimeChangedListener(times);
    }
    class OnChangeListener implements OnClickListener{
        @Override
        public void onClick(View v) {
            //TODO 自动存根法
            int h = timePick1.getCurrentHour();
            int m = timePick1.getCurrentMinute();
            System.out.println("h:" + h + "    m:" + m);
        }
    }
    class TimeListener implements OnTimeChangedListener{
        /**
```

```
         * view 为当前选中 TimePicker 控件
         * hourOfDay 为当前控件选中 TimePicker 的小时
         * minute 为当前选中控件 TimePicker 的分钟
         */
        @Override
        public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {
            //TODO 自动存根法
            System.out.println("h:" + hourOfDay + " m:" + minute);
        }
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

运行程序,效果如图 3-19 所示。



图 3-19 时间的设置

3.4 下拉列表控件

因为手机的屏幕较小,因此使用下拉列表来进行选择式输入是一个非常好的方式。Spinner 与 ListView 一样,也是 AdapterView 的一个间接子类,是一个显示数据的窗口。

1. Spinner 控件概述

Spinner 位于 android.widget 包下,每次只显示用户选中的元素,当用户再次单击时,会弹出选择列表供用户选择,而选择列表的元素同样来自适配器。图 3-20 所示为该类的继承树。

2. Spinner 控件经典案例

下面通过一个案例来演示 Spinner 控件的用法。

【例 3-16】 利用 Spinner 控件选择自己的爱好。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li3_16Spinner。

```
Java.lang.Object
└─Android.view.View
  └─Android.view.ViewGroup
    └─Android.widget.AdaptrView
      └─Android.widget.AbsSpinner
        └─Android.widget.Spinner
```

图 3-20　Spinner 类的继承树

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明 TextView 控件和一个 Spinner 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
android:orientation = "vertical"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent"
android:background = "#aabbcc" >
<TextView
    android:text = "@string/ys"
    android:id = "@ + id/TextView1"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:textSize = "28dip"/>
<Spinner
    android:id = "@ + id/Spinner1"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"/>
</LinearLayout>
```

(3) 打开 res\values 目录下的 string.xml 文件，为变量赋值，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <string name = "app_name">Spinner 案例</string>
    <string name = "action_settings">Settings</string>
    <string name = "hello_world">Hello world!</string>
    <string name = "ys">您的爱好</string>
    <string name = "lq">篮球</string>
    <string name = "zp">足球</string>
    <string name = "pq">排球</string>
</resources>
```

(4) 在 res\values 目录下创建一个颜色资源文件 color.xml，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
<color name = "red">#fd8d8d</color>
<color name = "green">#9cfda3</color>
<color name = "blue">#8d9dfd</color>
<color name = "white">#FFFFFF</color>
<color name = "black">#000000</color>
</resources>
```

(5) 打开 src\fs.li3_16spinner 包下的 MainActivity.java 文件，在文件中实现列表框的选择，其代码为：

```
package fs.li3_16spinner;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.AdapterView.OnItemSelectedListener;
public class MainActivity extends Activity {
    final static int WRAP_CONETNT = -2;          //表示 WRAP_CONTENT 的常量
    //所有资源的图片(足球、篮球、排球) id 的数组
    int[] drawableIds = { R.drawable.fb1, R.drawable.fb2,R.drawable.fb3 };
    //所有资源字符串 (足球、篮球、排球) id 的数组
    int[] msgIds = { R.string.zp, R.string.lq, R.string.pq };
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Spinner sp = (Spinner) findViewById(R.id.Spinner1);
        BaseAdapter ba = new BaseAdapter() {
            public int getCount() {
                //一共三个选项
                return 3;
                }
            public Object getItem(int position) {
                return null;
                }
            public long getItemId(int position) {
                return 0;
                }
            @SuppressLint("ResourceAsColor")
            public View getView(int position, View convertView, ViewGroup parent) {
            /* 动态生成每个下拉项对应的 View,每个下拉项 View 由 LinearLayout
                中包含一个 ImageView 及一个 TextView 构成 */
                //初始化 LinearLayout
                LinearLayout b = new LinearLayout(MainActivity.this);
                b.setOrientation(LinearLayout.HORIZONTAL);
                //初始化 ImageView
                ImageView ii = new ImageView(MainActivity.this);

                ii.setImageDrawable((getResources().getDrawable(drawableIds[position])));
                b.addView(ii);
                //初始化 TextView
                TextView tv = new TextView(MainActivity.this);
                tv.setText(" " + getResources().getText(msgIds[position]));
                tv.setTextColor(R.color.black);
                tv.setTextSize(24);
                b.addView(tv);
                return b;
                }
```

```
                };
                //为 Spinner 设置内容适配器
                sp.setAdapter(ba);
                sp.setOnItemSelectedListener(new OnItemSelectedListener() {
        public void onItemSelected(AdapterView<?> parent, View view,int position, long id) {
                    //获取主界面 TextView
                    TextView tv = (TextView) findViewById(R.id.TextView1);
                    //获取当前选中选项对应的 LinearLayout
                    LinearLayout ll = (LinearLayout) view;
                    //获取其中的 TextView
                    TextView tvn = (TextView) ll.getChildAt(1);
                    //用 StringBuilder 动态生成信息
                    StringBuilder sb = new StringBuilder();
                    sb.append(getResources().getText(R.string.ys));
                    sb.append(":");
                    sb.append(tvn.getText());
                    //信息设置进住界面
                    tv.setText(sb.toString());
                    }
                public void onNothingSelected(AdapterView<?> parent) {}
                });
            }
    }
```

运行程序,默认界面如图 3-21(a)所示,当单击列表框右侧的“三角”符号,弹出列表选择项,效果如图 3-21(b)所示,选择对应的喜好,即在文本框中显示对应的选项,如图 3-21(c)所示。



(a) 默认界面　　(b) 弹出列表选项　　(c) 选择选项

图 3-21　下拉列表控件效果

3.5　进度条控件

进度条(ProgressBar)类同样位于 android.widget 包下,但其继承自 View 类,主要用于显示一些操作的进度,应用程序可以修改其长度来表示当前后台操作的完成情况。因为进度条

会移动，所以长时间加载某些资源或执行某些耗时的操作时，不会使用户界面失去响应。

Android 当中的进度条 ProgressBar 有两种进度条，一种为垂直（圆圈）；另一种为水平（水平线）。

1. ProgressBar 控件概述

Android 支持几种风格的进度条，通过 style 属性可以为 ProgressBar 指定风格。该属性可支持以下几个属性值。

- @android:style/Widget.ProgressBar.Horizontal：水平进度条。
- @android:style/Widget.ProgressBar.Inverse：普通大小进度条。
- @android:style/Widget.ProgressBar.Large：大进度条。
- @android:style/Widget.ProgressBar.Large.Inverse：普通大进度条。
- @android:style/Widget.ProgressBar.Small：小进度条。
- @android:style/Widget.ProgressBar..Small.Inverse：普通小进度条。

另外，ProgressBar 还支持如表 3-8 所示的常用 XML 属性。

表 3-8 ProgressBar 常用的 XML 属性

XML 属性	描 述
android:max	设置该进度条的最大值
android:progress	设置该进度条的已完成进度值
android:progressDrawable	设置该进度条的轨道的绘制形式
android:indeterminate	该属性设为 true，设置进度条不精确显示进度
android:indeterminateDrawable	设置绘制不显示进度的进度条 Drawable 对象
android:indeterminateDuration	设置不精确显示进度的持续时间

表 3-8 中 android:progressDrawable 用于指定进度条轨道的绘制形式，该属性可指定为一个 LayerDrawable 对象（该对象可通过在 XML 文件中用＜layer-list＞元素进行配置）的引用。

ProgressBar 提供了如下方法来操作进度。

- setProgress(int)：设置进度的完成百分比。
- incrementProgressBy(int)：设置进度条的进度增加或减少。当参数为正数时进度增加；当参数为负数时进度减少。

2. ProgressBar 控件经典案例

下面通过一个案例来演示 ProgressBar 控件的用法。

【例 3-17】 利用 ProgressBar 控件实现两种类型进度条。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li3_17ProgressBar。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明 4 个 ProgressBar 控件，其代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
```

```
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc" >
    <!-- ProgressBar - 进度条控件 -->
    <!-- 以下分别为大、中、小的进度条控件(圆圈状) -->
    < ProgressBar
        android:id = "@ + android:id/progress_small"
        style = "?android:attr/progressBarStyleSmall"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content" />
    <!-- 进度条控件(条状)的演示.style 为进度条的样式,本例使用内置样式,max 为进度的最大值,
progress 为第一进度位置,secondaryProgress 为第二进度位置 -->
    < ProgressBar
        android:id = "@ + id/progress_horizontal"
        style = "?android:attr/progressBarStyleHorizontal"
        android:layout_width = "200px"
        android:layout_height = "wrap_content"
        android:layout_above = "@ + android:id/progress_large"
        android:layout_alignLeft = "@ + android:id/progress_small"
        android:max = "100"
        android:progress = "50"
        android:secondaryProgress = "75" />
    < ProgressBar
        android:id = "@ + android:id/progress_large"
        style = "?android:attr/progressBarStyleLarge"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + android:id/progress_small"
        android:layout_below = "@ + android:id/progress_small"
        android:layout_marginTop = "34dp" />
    < ProgressBar
        android:id = "@ + android:id/progress"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_below = "@ + android:id/progress_large"
        android:layout_marginTop = "31dp"
        android:layout_toRightOf = "@ + android:id/progress_small" />
</RelativeLayout >
```

(3) 打开 src\fs. li3_17progressbar 包下的 MainActivity. java 文件,在文件中实现长形进度条与圆形进度条的进度,其代码为:

```
package fs.li3_17progressbar;
import android.app.Activity;
import android.os.Bundle;
import android.view.Window;
//另见对话框中的进度条
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //TODO 自动存根法
        super.onCreate(savedInstanceState);
        //设置特性以允许在应用程序的标题栏上显示进度条(条状)
        requestWindowFeature(Window.FEATURE_PROGRESS);
        //设置特性以允许在应用程序的标题栏上显示进度条(圆圈状)
        requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
```

```
        this.setContentView(R.layout.main);
        setTitle("ProgressBar");
        //在标题栏上显示进度条(条状)
        setProgressBarVisibility(true);
        //在标题栏上显示进度条(圆圈状)
        setProgressBarIndeterminateVisibility(true);
        //指定进度条的进度
        setProgress(50 * 100);
        setSecondaryProgress(75 * 100);
        }
    }
```

运行程序,效果如图3-22所示。



图3-22 进度条

3.6 滑块控件

SeekBar(滑块)继承自ProgressBar,是用来接收用户输入的控件,类似于拖拉条,可以直观地显示用户需要的数据,常用于声音调节等场合。SeekBar不但可以直观地显示数值的大小,而且还可以为其设置标度,类似于显示在屏幕中的一把尺子。

1. SeekBar 控件属性

SeekBar允许用户改变拖动条的滑块外观,改变滑块外观通过如下属性来指定。

android:thumb:指定一个Drawable对象,该对象将作为自定义滑块。

Seekbar中还有几个重要的属性,分别为:

- android:layout_height="wrap_content":建议使用wrap_content,否则一定要保证设置的值不小于seekbar图片资源中的最高值。
- android:maxHeight="12px":说明进度条的最大高度。
- android:minHeight="12px":说明进度条的最低高度。
- android:paddingLeft="18px"或android:paddingRight="18px":解决拖动按钮在最左最右显示不全的问题,padding的值一般是thumb的一半宽度。

- android:progressDrawable="@drawable/seekbar_style"：设置了此值，就表示使用自定义的进度条样式，在其中可以设置进度条背景图，进度条图，缓冲条图。

为了让程序能响应 SeekBar 滑块位置的改变，程序可以考虑为它绑定一个 OnSeekBarChangeListener 监听器。

2. SeekBar 控件经典案例

下面通过一个案例来演示 SeekBar 控件的用法。

【例 3-18】 本案例演示：Activity 上有一个 SeekBar 和一个 TextView，当拖动 SeekBar 的进度时，在下面的 TextView 中显示相应的进度变化。其具体操作步骤如下。

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li3_18SeekBar。

(2) 打开 res\layout 目录下的 main. xml 布局文件，在文件中声明一个 SeekBar 控件和两个 TextView 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc" >
< SeekBar
     android:id = "@ + id/seekbar"
     android:layout_width = "fill_parent"
     android:layout_height = "wrap_content">
</SeekBar >
< EditText
     android:id = "@ + id/edit"
     android:layout_width = "127dp"
     android:layout_height = "wrap_content"
     android:ems = "10"
     android:text = "当前值为: " />
< EditText
     android:id = "@ + id/edit2"
     android:layout_width = "wrap_content"
     android:layout_height = "wrap_content"
     android:layout_gravity = "right"
     android:ems = "10" >
  </EditText >
</LinearLayout >
```

(3) 打开 src\fs. li3_18seekbar 包下的 MainActivity. java 文件，在文件中实现当拖动 SeekBar 控件时，即在将对应的值显示在 TextView 中，其代码为：

```
package fs.li3_18seekbar;
import android.app.Activity;
import android.os.Bundle;
import android.widget.SeekBar;
import android.widget.SeekBar.OnSeekBarChangeListener;
import android.widget.TextView;
public class MainActivity extends Activity {
    //第一次调用活动
    //定义一个 SeekBar 和一个 TextView
    private SeekBar seekBar;
    private TextView  textView;
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //根据 ID 值取得 SeekBar 对象
        seekBar = (SeekBar)findViewById(R.id.seekbar);
        seekBar.setMax(100);
       //为 SeekBar 设置监听器(这里使用匿名内部类)
        seekBar.setOnSeekBarChangeListener(new OnSeekBarChangeListener(){
        //复写 OnSeeBarChangeListener 的三个方法
        //第一个是 OnStartTrackingTouch,在进度开始改变时执行
        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {
        //TODO 自动存根法
        }
        //第二个方法 onProgressChanged 是当进度发生改变时执行
        @Override
      public void onProgressChanged(SeekBar seekBar, int progress,boolean fromUser) {
            //TODO 自动存根法
            textView = (TextView)findViewById(R.id.edit2);
            int i = seekBar.getProgress();
            textView.setText("" + i);
            }
        //第三个是 onStopTrackingTouch,在停止拖动时执行
        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
        //TODO 自动存根法
        textView = (TextView)findViewById(R.id.edit2);
        int i = seekBar.getProgress();
        textView.setText("" + i);
        }
     });
    }
}
```

运行程序,效果如图 3-23(a)所示,当拖动 SeekBar 控件时,即在 TextView 显示对应的值,效果如图 3-23(b)所示。



(a) 默认界面



(b) 拖动SeekBar界面

图 3-23　滑块控件的使用

3.7 星级滑块控件

前面已经介绍过普通滑块的使用，本节将介绍另一种滑块——星级滑块(RatingBar)，该控件的使用较少，一般用于星级评分的场合。

1. RatingBar 控件概述

RatingBar 是基于 SeekBar(拖动条)和 ProgressBar(状态条)的扩展，用星形来显示等级评定，在使用默认 RatingBar 时，用户可以通过触摸/拖动/按键(如遥控器)来设置评分，RatingBar 自带两种模式，一个小风格 ratingBarStyleSmall；大风格 ratingBarStyleIndicator，大风格只适合做指示，不适用与用户交互。

表 3-9 列出了 RatingBar 所支持的常见 XML 属性。

表 3-9 RatingBar 支持的常见 XML 属性

XML 属性	描 述
android:isIndicator	设置该星级评分条是否允许用户改变(true 为不允许修改)
android:numStars	设置该星级评分条总共有多少个星级
android:rating	设置该星级评分条默认的星级
android:stepSize	设置每次最少需要改变多少个星级

2. RatingBar 控件经典案例

下面通过一个案例来演示 RatingBar 控件的使用。

【例 3-19】 使用 RatingBar 控件实现星级评分。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li3_19RatingBar。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中实现 3 个线性布局，并声明 4 个 RatingBar 控件及 1 个 TextView 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:paddingLeft = "10dip"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:background = "#aabbcc" >
    < RatingBar
        android:id = "@ + id/ratingbar1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:numStars = "3"
        android:rating = "2.5" />
    < RatingBar
        android:id = "@ + id/ratingbar2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:numStars = "5"
        android:rating = "2.25" />
    < LinearLayout
```

```
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:layout_marginTop = "10dip">
            < TextView
                android:id = "@ + id/rating"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content" />
            < RatingBar
                android:id = "@ + id/small_ratingbar"
                style = "?android:attr/ratingBarStyleSmall"
                android:layout_marginLeft = "5dip"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:layout_gravity = "center_vertical" />
        </LinearLayout >
        < RatingBar
            android:id = "@ + id/indicator_ratingbar"
            style = "?android:attr/ratingBarStyleIndicator"
            android:layout_marginLeft = "5dip"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_gravity = "center_vertical" />
    </LinearLayout >
```

(3) 打开 src\fs. li3_19ratingbar 包下的 MainActivity. java 文件,在文件中实现星级滑块的评分,并将相应评分显示在 TextView 控件中,其代码为:

```
package fs.li3_19ratingbar;
import android.app.Activity;
import android.os.Bundle;
import android.widget.RatingBar;
import android.widget.TextView;
import android.widget.RatingBar.OnRatingBarChangeListener;
public class MainActivity extends Activity implements OnRatingBarChangeListener {
    private RatingBar mSmallRatingBar;
    private RatingBar mIndicatorRatingBar;
    private TextView mRatingText;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //TODO 自动存根法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mRatingText = (TextView) findViewById(R.id.rating);
        mIndicatorRatingBar = (RatingBar) findViewById(R.id.indicator_ratingbar);
        mSmallRatingBar = (RatingBar) findViewById(R.id.small_ratingbar);
         //根据监听在布局文件中布局不同的滑块
       ((RatingBar)findViewById(R.id.ratingbar1)).setOnRatingBarChangeListener(this);
       ((RatingBar)findViewById(R.id.ratingbar2)).setOnRatingBarChangeListener(this);
         }
    @Override
    public void onRatingChanged(RatingBar ratingBar, float rating,boolean fromUser) {
        final int numStars = ratingBar.getNumStars();
        mRatingText.setText(" 受欢迎度" + rating + "/" + numStars);
```

```
            if (mIndicatorRatingBar.getNumStars()!= numStars) {
                mIndicatorRatingBar.setNumStars(numStars);
                mSmallRatingBar.setNumStars(numStars);
            }
            if (mIndicatorRatingBar.getRating()!= rating) {
                mIndicatorRatingBar.setRating(rating);
                mSmallRatingBar.setRating(rating);
            }
            final float ratingBarStepSize = ratingBar.getStepSize();
            if (mIndicatorRatingBar.getStepSize()!= ratingBarStepSize) {
                mIndicatorRatingBar.setStepSize(ratingBarStepSize);
                mSmallRatingBar.setStepSize(ratingBarStepSize);
            }
        }
    }
```

运行程序,效果如图 3-24 所示。



图 3-24　星级滑块评分

3.8　综合经典案例

前面内容已对 Android 的控件作了基本介绍,并都给出相应的案例来进行说明,本节将通过一个综合案例来总结 Android 控件的用法。

【例 3-20】　本案例要演示的是完成一个 Android 中的 fsComeMini 登录界面。

下面将分步骤讲解怎样实现相应的界面效果,让大家都能轻松的做出美观的登录界面。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li3_20Integrate。

(2) 打开 res\values 目录下的 strings. xml 文件,在文件中对一些字符进行定义,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "app_name">控件综合案例</string >
    < string name = "action_settings"> Settings </string >
    < string name = "hello_world"> Hello world! </string >
    < string name = "login_label_username">账号</string >
    < string name = "login_label_password">密码</string >
    < string name = "login_label_signin">登 录</string >
    < string name = "login_status_logging_in">登录中...</string >
    < string name = "login_username_hint"> E - mail 或手机号</string >
    < string name = "login_register_link">没有账号? < a href = "#" mce_href = "#">注册</a ></string >
</resources >
```

(3) 在 res\values 中创建一个 style.xml 文件,在文件中对这个 TextView 设置字体颜色和字体大小,其代码为:

```
< resources >
        < style name = "normalText" parent = "@android:style/TextAppearance">
        < item name = "android:textColor"> #444 </item >
        < item name = "android:textSize"> 14sp </item >
        </style >
</resources >
```

(4) 打开 res\drawable-mdpi 下,在该目录下放置两个图片文件,分为 q1.jpg 及 c8.jpg。

(5) 在 res\drawable-mdpi 下创建一个 backcolor.xml 文件,用于设置背景色为渐变色,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< shape xmlns:android = "http://schemas.android.com/apk/res/android">
      < gradient
           android:startColor = " #FFACDAE5"
           android:endColor = " #FF72CAE1"
           android:angle = "45"/>
</shape >
```

(6) 在 res\drawable-mdpi 下创建一个 blackcolorfillet.xml 文件,用于设置背景色为淡蓝色且为圆角,其代码为:

```
< shape xmlns:android = "http://schemas.android.com/apk/res/android">
        < solid android:color = " #55FFFFFF" />
        <!-- 设置圆角,表示的是一个带圆角且背景色为#55FFFFFF(淡蓝色)
      注意 bottomRightRadius 是左下角而不是右下角   bottomLeftRadius 右下角 -->
        < corners
            android:topLeftRadius = "10dp"
            android:topRightRadius = "10dp"
            android:bottomRightRadius = "10dp"
            android:bottomLeftRadius = "10dp"/>
</shape >
```

(7) 打开 res\layout 目录下的 main.xml 布局文件,在文件中实现一个 LinearLayout 布局,两个 RelativeLayout 布局,并声明 3 个 TextView 控件、2 个 EditText 控件、2 个 ImageButton 控件及 1 个 Button 控件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
```

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:background="@drawable/backcolor">
  <!-- padding 为内边距; layout_margin 为外边距; android:layout_alignParentTop 为布局的位置是
否处于顶部 -->
  <RelativeLayout
          android:id="@+id/login_div"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:padding="15dip"
          android:layout_margin="15dip"
          android:background="@drawable/blackcolorfillet">
          <!-- 账号 -->
          <TextView
                  android:id="@+id/login_user_input"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:layout_alignParentTop="true"
                  android:layout_marginTop="5dp"
                  android:text="@string/login_label_username"
                  style="@style/normalText"/>
          <EditText
                  android:id="@+id/username_edit"
                  android:layout_width="fill_parent"
                  android:layout_height="wrap_content"
                  android:hint="@string/login_username_hint"
                  android:layout_below="@id/login_user_input"
                  android:singleLine="true"
                  android:inputType="text"/>
    <!-- 密码 text -->
    <TextView
            android:id="@+id/login_password_input"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/username_edit"
            android:layout_marginTop="3dp"
            android:text="@string/login_label_password"
            style="@style/normalText"/>
    <EditText
            android:id="@+id/password_edit"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_below="@id/login_password_input"
            android:password="true"
            android:singleLine="true"
            android:inputType="textPassword"/>
    <!-- 登录 button -->
    <Button
            android:id="@+id/signin_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/password_edit"
```

```
            android:layout_alignRight = "@id/password_edit"
            android:text = "@string/login_label_signin"
            android:background = "#00ffff"/>
  </RelativeLayout>
  <RelativeLayout
      android:layout_width = "fill_parent"
      android:layout_height = "wrap_content"
      android:background = "#aabbcc" >
         <TextView
             android:id = "@+id/register_link"
             android:text = "@string/login_register_link"
             android:layout_width = "wrap_content"
             android:layout_height = "wrap_content"
             android:layout_marginLeft = "15dp"
             android:textColor = "#888"
             android:textColorLink = "#FF0066CC"/>
            <ImageView
                android:id = "@+id/logo"
                android:src = "@drawable/c8"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:layout_alignParentRight = "true"
                android:layout_alignParentBottom = "true"
                android:layout_marginRight = "25dp"
                android:layout_marginLeft = "10dp"
                android:layout_marginBottom = "25dp" />
         <ImageView
             android:src = "@drawable/q1"
             android:layout_width = "wrap_content"
             android:layout_height = "wrap_content"
             android:layout_toLeftOf = "@id/logo"
             android:layout_alignBottom = "@id/logo"
             android:paddingBottom = "8dp"/>
            </RelativeLayout>
</LinearLayout>
```

(8) 打开 src\fs.li3_20integrate 包下的 MainActivity.java 文件，实现启动 Activity，其代码为：

```
package fs.li3_20integrate;
import android.app.Activity;
import android.os.Bundle;
import android.view.Window;
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.main);
    }
}
```

注意：该 Activity 是无标题的，需要在 setContentView 之前设置无标题，否则会报错。

运行程序，效果如图 3-25 所示。



图 3-25　fsComeMini 登录界面

第 4 章 Android 视图

视图(View)是一个矩形区域,负责这个区域里的绘制和事件处理。视图类是 Android 用户界面的基础类之一。视图组(ViewGroup)是视图的子类,是一个容器,专门负责布局。视图组本身没有可绘制的元素。

4.1 滚动视图

滚动视图(ScrollView)是当需要显示的信息一个屏幕显示不下时使用的控件。

1. ScrollView 概述

ScrollView 由 FrameLayout 派生,同样位于 android. widget 包下。ScrollView 类实际上是一个帧布局,一般情况下,其中的控件是按照线性进行布局的,用户可以对其进行滚动,以达到在屏幕中显示更多信息的目的。

默认情况下,ScrollView 只是为其他组件添加垂直滚动条,如果应用需要添加水平滚动条,则可借助于另一个滚动条视图来实现。ScrollView 与 HorizontalScrollView 的功能基本相似,只是前者添加垂直滚动条;后者添加水平滚动条。

ScrollView 的使用与普通布局的使用没有太大区别,可以在 XML 文件中进行配置,也可以通过 Java 代码进行设置。在 ScrollView 中可以添加任意满足条件的控件,当一个屏幕显示不下其中所包含的所有控件或信息时,便会自动添加滚动功能。

注意:ScrollView 中同一时刻只能包含一个 View。

2. 在 XML 文件中配置 ScrollView 控件

下面通过一个示例来演示在 XML 文件中配置 ScrollView 控件。

【例 4-1】 利用 ScrollView 类完成图片的垂直滚动。其实现的具体步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li4_1ScrollViewXML。

(2) 打开 res\layout 目录下的 main. xml 文件,在文件中布局图像的垂直滚动,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< ScrollView xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:scrollbars = "vertical"
    android:background = "#000000">
    < LinearLayout
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent">
        < ImageView
            android:layout_width = "wrap_content"
```

```
                android:layout_height = "wrap_content"
                android:src = "@drawable/d5"
                android:layout_gravity = "center_horizontal"/>
            < ImageView
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/d5"
                android:layout_gravity = "center_horizontal"/>
            < ImageView
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/d5"
                android:layout_gravity = "center_horizontal"/>
            < ImageView
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/d5"
                android:layout_gravity = "center_horizontal"/>
            < ImageView
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/d5"
                android:layout_gravity = "center_horizontal"/>
            < ImageView
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/d5"
                android:layout_gravity = "center_horizontal"/>
            < ImageView
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/d5"
                android:layout_gravity = "center_horizontal"/>
            < ImageView
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/d5"
                android:layout_gravity = "center_horizontal"/>
            < ImageView
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/d5"
                android:layout_gravity = "center_horizontal"/>
        </LinearLayout >
    </ScrollView >
```

运行程序,效果如图 4-1 所示。

3. 在 Java 文件中配置 ScrollView 控件

ScrollView 卷轴视图是指当拥有很多内容,一屏显示不完时,需要通过滚动条来显示视图的使用。下面通过一个案例来演示在 Java 文件中配置 ScrollView 控件。

【例 4-2】 实现在 Java 代码中设置 ScrollView 控件。其具体操作步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li4_2ScrollViewJava。

(2) 打开 res\layout 目录下的 main.xml 布局文件,在文件中声明一个 ScrollView 控件及



图 4-1　图片的垂直滚动

一个 TextView 控件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#000000">
<!-- ScrollView 为滚动条控件; scrollbarStyle 为滚动条的样式 -->
< ScrollView
     android:id = "@ + id/scrollView"
     android:layout_width = "fill_parent"
     android:layout_height = "200px"
     android:scrollbarStyle = "outsideOverlay"
     android:background = "#aabbcc">
< TextView
     android:layout_width = "fill_parent"
     android:layout_height = "wrap_content"
     android:id = "@ + id/textView" />
</ScrollView >
</LinearLayout >
```

(3) 打开 src\fs. li4_2scrollviewjava 包下的 MainActivity. java 文件,在文件中实现 ScrollView 的设置,其代码为:

```
package fs.li4_2scrollviewjava;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        //TODO 自动存根法
        super.onCreate(savedInstanceState);
        this.setContentView(R.layout.main);
        setTitle("ScrollView");
        TextView textView = (TextView)this.findViewById(R.id.textView);
        textView.setText("我是 Android\n 我是 Android\n 我是 Android\n 我是 Android\n 我是
Android\n 我是 Android\n 我是 Android\n 我是 Android\n 我是 Android\n 我是 Android\n 我是 Android\
n 我是 Android\n 我是 Android\n 我是 Android\n 我是 Android\n 我是 Android\n 我是 Android\n 我是
Android\n 我是 Android\n 我是 Android\n 我是 Android\n 我是 Android\n 我是 Android\n 我是 Android\
n 我是 Android\n 我是 Android\n 我是 Android\n 我是 Android ");
        }
}
```

运行程序，效果如图 4-2 所示。



图 4-2　在 Java 设置 ScrollView

4.2　图 片 控 件

本节将要介绍的是图片控件 ImageView，首先对 ImageView 类进行简单介绍，然后通过一个案例来说明 ImageView 的用法。

1. ImageView 控件概述

图像视图使用 ImageView 表示，用于在屏幕中显示 Drawable 对象，通常用来显示图片。在 Android 中，可以使用两种方法向屏幕中添加图像视图，一种是通过在 XML 布局文件中使用<ImageView>标记添加；另一种是在 Java 文件中通过 new 关键字创建。推荐采用第一种方法。

在使用 ImageView 组件显示图像时，通常可将要显示的图片放置在 res\drawable 目录中，然后用以下代码将其显示在布局管理器中。

```
< ImageView
    属性列表>
</ ImageView >
```

表 4-1 显示了 ImageView 支持的常用 XML 属性及方法描述。

表 4-1　ImageView 支持的 XML 属性及方法描述

XML 属性	方　法	描　述
android:adjustViewBounds	setAdjustViewBounds(boolean)	设置 ImageView 是否调整自己的边界来保持所显示图片的长宽比
android:maxHeight	setMaxHeight(int)	设置 ImageView 的最大高度
android:maxWidth	setMaxWidth(int)	设置 ImageView 的最大宽度
android:scaleType	setScaleType(ImageView.ScaleType)	设置所显示的图片如何缩放或移动以适应 ImageView 的大小
android:src	setImageResource(int)	设置 ImageView 所显示的 Drawable 对象的 ID

通过 android:src 属性设置 ImageView 显示图片，也可显示颜色。

同时 ImageView 类中还有一些常用的成员方法，如表 4-2 所示。

表 4-2　ImageView 中常用方法及说明

名　称	说　明
setAlpha(int alpha)	设置 ImageView 的透明度
setImageBitmap(Bitmap bm)	设置 ImageView 所显示的内容为指定的 Bitmap 对象
setImageDrawable(Drawable drawble)	设置 ImageView 所显示的内容为指定 id 的资源
setImageResource(int resId)	设置 ImageView 所显示的内容为指定 URI
setSelected(boolean selected)	设置 ImageView 的选中状态

2. ImageView 控件经典案例

下面通过一个案例来演示 ImageView 的用法。

【例 4-3】 使用 ImageView 控件演示图片的效果并查看图片。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_3ImageView。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中定义 2 个线性布局，并声明 1 个 ImageView 控件及 4 个 Button 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = " # aabbcc">
  < ImageView
            android:id = "@ + id/imageView"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_gravity = "center_horizontal"
            android:src = "@drawable/face"/>
< LinearLayout
            android:orientation = "horizontal"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:layout_gravity = "center_horizontal"
            android:background = " # aabbcc">
< Button
     android:id = "@ + id/alpha_plus"
```

```
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "透明度增加"
            android:layout_gravity = "center_horizontal"/>
    < Button
            android:id = "@ + id/alpha_minus"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "透明度减少" />
      </LinearLayout >
    < Button
            android:id = "@ + id/next"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "下一张" />
    < Button
            android:id = "@ + id/previous"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "上一张" />
    </LinearLayout >
```

(3) 打开 src\fs. li4_3imageview 包下的 MainActivity. java 文件，在文件中实现图片的浏览与改变图像的透明度，其代码为：

```
package fs.li4_3imageview;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
public class MainActivity extends Activity {
    private ImageView imageView = null;
    private Button previous = null;
    //上一张
    private Button next = null;
    //下一张
    private Button alpha_plus = null;
    //透明度增加
    private Button alpha_minus = null;
    //透明度减少
    private int currentImgId = 0;
    //记录当前 ImageView 显示的图片 id
    private int alpha = 255;
    //记录 ImageView 的透明度
    int [] imgId = {                //ImageView 显示的图片数组
            R.drawable.c1,
            R.drawable.c2,
            R.drawable.c3,
            R.drawable.c4,
            R.drawable.c5,
            R.drawable.g4,
            R.drawable.g5,
            R.drawable.c8,
            };
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.main);
        imageView = (ImageView)findViewById(R.id.imageView);
        previous = (Button)findViewById(R.id.previous);
        next = (Button)findViewById(R.id.next);
        alpha_plus = (Button)findViewById(R.id.alpha_plus);
        alpha_minus = (Button)findViewById(R.id.alpha_minus);
        previous.setOnClickListener(listener);
        next.setOnClickListener(listener);
        alpha_plus.setOnClickListener(listener);
        alpha_minus.setOnClickListener(listener);
        }
    private View.OnClickListener listener = new View.OnClickListener(){
        public void onClick(View v) {
            if(v == previous){
                currentImgId = (currentImgId - 1 + imgId.length) % imgId.length;
                imageView.setImageResource(imgId[currentImgId]);
                }
            if(v == next){
                currentImgId = (currentImgId + 1) % imgId.length;
                imageView.setImageResource(imgId[currentImgId]);
                }
            if(v == alpha_plus){
                alpha += 10;
                if(alpha > 255){
                    alpha = 255;
                    }
                imageView.setAlpha(alpha);
                }
            if(v == alpha_minus){
                alpha -= 10;
                if(alpha < 0){
                    alpha = 0;
                    }
                imageView.setAlpha(alpha);
                }
            }
        };
}
```

运行程序,效果如图 4-3 所示。

图 4-3　浏览图片

4.3　列 表 视 图

在 Android 中 ListView 是常用的控件,根据列表的适配器类型,列表分为 ArrayAdapter、SimpleAdapter 和 SimpleCursorAdapter 三种。

其中,ArrayAdapter 最为简单,只能展示一行字;SimpleAdapter 有最好的扩充性,可以自定义出各种效果;SimpleCursorAdapter 可以认为是 SimpleAdapter 对数据库的简单结合,可以方便把数据库的内容以列表的形式展示出来。

1. ListView 控件概述

ListView 类位于 android. widget 包下，该类的使用方法非常简单，只需先初始化所需要的数据，然后创建适配器并将其设置给 ListView。ListView 便将信息以列表的形式显示到页面中，该类的继承如图 4-4 所示。

2. ListView 控件经典案例

下面通过一个案例来演示 ListView 控件的用法。

【例 4-4】 利用 ListView 控件实现数据的显示。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_4ListView。

```
Java.lang.Object
  └─Android.view.View
     └─Android.view.ViewGroup
        └─Android.widget.AdapterView
           └─Android.widget.AbsListView
              └─Android.widget.ListView
```

图 4-4 ListView 类的继承树

(2) 打开 res\layout 目录下的 main. xml 布局文件，在文件中声明一个 ListView 控件。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/LinearLayout01"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = " # aabbcc">
    < ListView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:id = "@ + id/MyListView">
    </ListView >
</LinearLayout >
```

(3) 在 res\values 目录中创建一个 listitem. xml 文件，在文件中布局两个 TextView 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout android:layout_width = "fill_parent"
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_height = "wrap_content"
    android:id = "@ + id/myListItem"
    android:paddingBottom = "3dip"
    android:paddingLeft = "10dip"
    android:background = " # aabbcc">
    < TextView android:layout_height = "wrap_content"
        android:layout_width = "fill_parent"
        android:id = "@ + id/itemTitle"
        android:textSize = "20dip">
    </TextView >
    < TextView android:layout_height = "wrap_content"
        android:layout_width = "fill_parent"
        android:id = "@ + id/itemText">
    </TextView >
</LinearLayout >
```

(4) 打开 src\fs. li4_4listview 包下的 MainActivity. java 文件，在文件中实现字符串显示，其代码为：

```
package fs.li4_4listview;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ListView;
import android.widget.SimpleAdapter;
public class MainActivity extends Activity {
    private ListView list = null;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        list = (ListView) findViewById(R.id.MyListView);
        //组织数据源
        List<HashMap<String, String>> mylist = new ArrayList<HashMap<String, String>>();
        for(int i = 0;i<10;i++) {
            HashMap<String, String> map = new HashMap<String, String>();
            map.put("itemTitle", "This is Title");
            map.put("itemText", "This is text");
            mylist.add(map);
        }
        //配置适配器
        SimpleAdapter adapter = new SimpleAdapter(this,
                        mylist,                     //数据源
                        R.layout.listitem,          //显示布局
                        new String[] {"itemTitle", "itemText"},    //数据源的属性字段
                  new int[] {R.id.itemTitle,R.id.itemText});       //布局里的控件 id
        //添加并且显示
        list.setAdapter(adapter);
    }
}
```

运行程序,效果如图 4-5 所示。

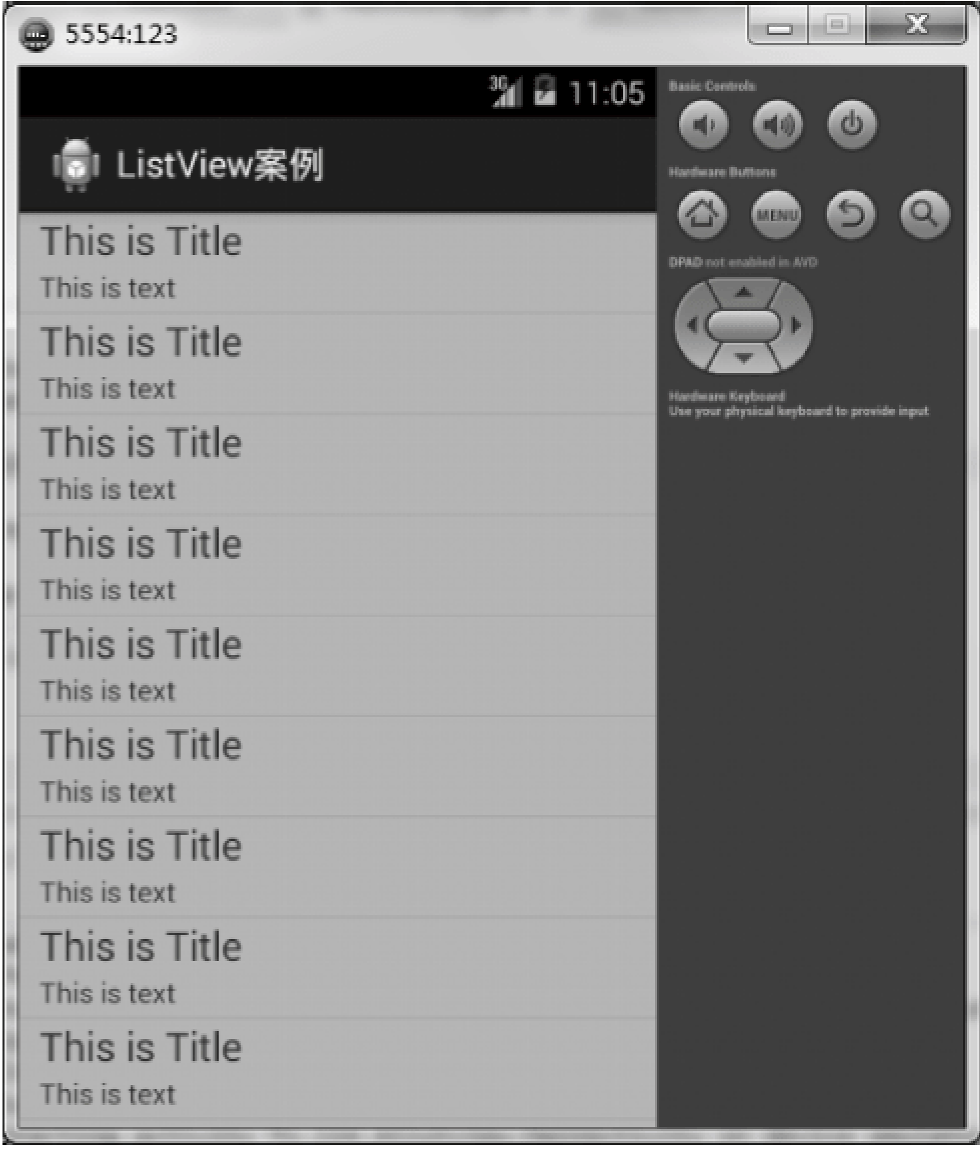


图 4-5　ListView 浏览字串

【例 4-5】 本案例通过 ListView 控件实现一个简单的名人录，其中包括名人的照片以及描述。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_6ListViewimage。

(2) 打开 res\layout 目录下的 main. xml 文件，在文件中定义了一个 TextView 控件和一个 ListView 控件，其代码为：

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#000000"
    android:orientation = "vertical" >
    <TextView
        android:id = "@ + id/TextView1"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:textColor = "@color/white"
        android:textSize = "24dip" />
    <ListView
        android:id = "@ + id/ListView1"
        android:layout_width = "fill_parent"
        android:layout_height = "200dp"
        android:layout_weight = "0.70"
        android:choiceMode = "singleChoice" >
    </ListView>
</LinearLayout>
```

(3) 打开 res\values 目录下的 strings. xml 文件，在文件中定义字串，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <string name = "app_name">ListView2 案例</string>
    <string name = "action_settings">Settings</string>
    <string name = "a">杀生丸</string>
    <string name = "b">宇智波佐助</string>
    <string name = "c">江户川柯南</string>
    <string name = "d">流川枫</string>
    <string name = "e">樱木花道</string>
    <string name = "ys">您选择了</string>
    <!-- 定义 ys 字符串 -->
</resources>
```

(4) 在 res\values 中创建一个颜色资源文件 color. xml 文件，用于设置文本框颜色，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <color name = "red">#fd8d8d</color>
    <color name = "green">#9cfda3</color>
    <!-- 声明 green 颜色 -->
    <color name = "blue">#8d9dfd</color>
    <!-- 声明 blue 颜色 -->
    <color name = "white">#FFFFFF</color>
    <!-- 声明 white 颜色 -->
    <color name = "black">#000000</color>
    <!-- 声明 black 颜色 -->
    <color name = "gray">#050505</color>
    <!-- 声明 gray 颜色 -->
</resources>
```

(5) 打开 src\fs. li4_6listviewimage 包下的 MainActivity. java 文件,在文件中实现 ListView 名人录功能,其代码为:

```
package fs.li4_6listviewimage;
import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;
public class MainActivity extends Activity {
    //所有资源图片 id 的数组
    int[] drawableIds = {R.drawable.po1,R.drawable.po2,R.drawable.po3,R.drawable.po4,
R.drawable.po5};
            //所有资源字符串 id 的数组
            int[] msgIds = {R.string.a,R.string.b,R.string.c,R.string.d,R.string.e};
           public void onCreate(Bundle savedInstanceState) {   //重写的 onCreate 方法
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);                 //设置当前的用户界面
           ListView lv = (ListView)this.findViewById(R.id.ListView1);         //初始化 ListView
           BaseAdapter ba = new BaseAdapter(){                 //为 ListView 准备内容适配器
        public int getCount() {return 5;}                      //总共 5 个选项
        public Object getItem(int arg0) { return null; }       //重写的 getItem 方法
        public long getItemId(int arg0) { return 0; }          //重写的 getItemId 方法
        public View getView(int arg0, View arg1, ViewGroup arg2) {
            //动态生成每个下拉项对应的 View,每个下拉项 View 由 LinearLayout
            //中包含一个 ImageView 及一个 TextView 构成
        LinearLayout ll = new LinearLayout(MainActivity.this); //初始化 LinearLayout
            ll.setOrientation(LinearLayout.HORIZONTAL);        //设置朝向
            ll.setPadding(5,5,5,5);                            //设置四周留白
        ImageView   ii = new ImageView(MainActivity.this);     //初始化 ImageView
        ii.setImageDrawable(getResources().getDrawable(drawableIds[arg0]));       //设置图片
            ii.setScaleType(ImageView.ScaleType.FIT_XY);
            ii.setLayoutParams(new Gallery.LayoutParams(100,98));
            ll.addView(ii);                                    //添加到 LinearLayout 中
            TextView tv = new TextView(MainActivity.this);     //初始化 TextView
            tv.setText(getResources().getText(msgIds[arg0]));//设置内容
            tv.setTextSize(24);                                //设置字体大小
          tv.setTextColor(MainActivity.this.getResources().getColor(R.color.white));  //字体颜色
                        tv.setPadding(5,5,5,5);                //设置四周留白
                        tv.setGravity(Gravity.LEFT);
                        ll.addView(tv);                        //添加到 LinearLayout 中
                        return ll;
                    }
                };
            lv.setAdapter(ba);                                 //为 ListView 设置内容适配器
                lv.setOnItemSelectedListener(                  //设置选项选中的监听器
```

```
                    new OnItemSelectedListener(){
                     public void onItemSelected(AdapterView<?> arg0, View arg1,
                 int arg2, long arg3) {                        //重写选项被选中事件的处理方法
        LinearLayout ll = (LinearLayout)arg1;               //获取当前选中选项对应的 LinearLayout
                TextView tvn = (TextView)ll.getChildAt(1);          //获取其中的 TextView
                StringBuilder sb = new StringBuilder();             //用 StringBuilder 动态生成信息
                      sb.append(getResources().getText(R.string.ys));
                      sb.append(":");
                      sb.append(tvn.getText());
                      String stemp = sb.toString();
                    }
                 public void onNothingSelected(AdapterView<?> arg0){}
                   }
                 );
                lv.setOnItemClickListener(                          //设置选项被单击的监听器
                    new OnItemClickListener(){
                 public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                       long arg3) {                                  //重写选项被单击事件的处理方法
          TextView tv = (TextView)findViewById(R.id.TextView1);  //获取主界面 TextView
        LinearLayout ll = (LinearLayout)arg1;               //获取当前选中选项对应的 LinearLayout
          TextView tvn = (TextView)ll.getChildAt(1);                //获取其中的 TextView
          StringBuilder sb = new StringBuilder();                   //用 StringBuilder 动态生成信息
            sb.append(getResources().getText(R.string.ys));
            sb.append(":");                                         //添加一个冒号
            sb.append(tvn.getText());                               //得到文本
            String stemp = sb.toString();
            tv.setText(stemp.split("\\n")[0]);    //信息设置进主界面 TextView
                    }
              });
          }
     }
```

运行程序，即在界面中显示图片，当选中其中某人物时，即其对应的信息显示在 TextView 控件中，如图 4-6 所示。



图 4-6　ListView 浏览图片

4.4 网格视图

本节将介绍另一种视图——网格视图(GridView),网格视图也是在应用程序中比较常见的视图。

1. GridView 控件概述

GridView 用于在界面上按行、列分布的方式来显示多个组件。

GridView 和 ListView 有共同的父类 AbsListView,因此 GridView 和 ListView 具有一定的相似性。GridView 与 ListView 的主要区别在于,ListView 只是在一个方向上分布;而 GridView 则会在两个方向上分布。

与 ListView 类似的是,GridView 也需要通过 Adapter 来提供显示的数据:开发者既可通过 SimpleAdapter 来为 GridView 提供数据,也可通过开发 BaseAdapter 的子类来为 GridView 提供数据。不管使用哪种方式,GridView 与 ListView 的用法基本是一致的。

GridView 提供如表 4-3 所示的常用 XML 属性。

表 4-3 GridView 常用的 XML 属性

XML 属性	方　法	描　述
android:columnWidth	setColumnWidth(int)	设置列的宽度
android:gravity	setGravity(int)	设置对齐方式
android:horizontalSpacing	setHorizontalSpacing(int)	设置各元素之间的水平间距
android:numColumns	setNumColumns(int)	设置列数
android:stretchMode	setStretchMode(int)	设置拉伸模式
android:verticalSpacing	setVerticalSpacing(int)	设置各元素之间的垂直间距

注意:使用 GridView 时一般都应该指定 numColumns 大于 1,否则该属性的默认值为 1。如果将属性设为 1,则意味着该 GridView 只有一列,那 GridView 就变成了 ListView。

android:stretchMode 属性支持如下几个属性值。

- NO_STRETCH:不拉伸。
- STRETCH_SPACING:仅拉伸元素之间的间距。
- STRETCH_SPACING_UNIFORM:表格元素本身、元素之间的间距一起拉伸。
- STRETCH_COLUMN_WIDTH:仅拉伸元素表格元素本身。

2. GridView 控件经典案例

下面通过一个案例来演示 GridView 控件的用法。

【例 4-6】 以下利用 GridView 控件模拟九宫图的实现,当鼠标点击图片时会进行相应的跳转链接。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li4_7GridView。

(2) 打开 res\layout 目录下的 main.xml 布局文件,在文件中声明一个 GridView 控件及两个 TextView 控件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
```

```
        android:layout_height = "fill_parent"
        android:background = "#aabbcc">
<TextView
    android:text = "这次决斗谁赢了!"
    android:id = "@+id/textView1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"/>
<GridView
    android:layout_height = "wrap_content"
    android:id = "@+id/gridView"
    android:layout_width = "match_parent"
    android:numColumns = "2"
    android:background = "#FFF"/>
<TextView
     android:layout_height = "wrap_content"
     android:layout_width = "fill_parent"
     android:text = "@string/hello_world"
     android:id = "@+id/text"/>
</LinearLayout>
```

(3) 在 res\layout 目录下创建一个 item.xml 文件,用于为每个 grid 的单项设置一个样式,其代码为:

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
        android:orientation = "vertical"
        android:layout_width = "match_parent"
        android:layout_height = "match_parent"
        android:background = "#aabbcc">
<ImageView
    android:layout_height = "wrap_content"
    android:id = "@+id/item_imageView"
    android:layout_width = "wrap_content"
    android:src = "@drawable/po1"
    android:layout_gravity = "center"/>
<TextView
    android:text = "TextView"
    android:id = "@+id/item_textView"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_gravity = "center"/>
</LinearLayout>
```

(4) 打开 src\fs.li4_7gridview 包下的 MainActivity.java 文件,在文件中实现当单击相关图片时,即把相关信息显示在 TextView 控件中,其代码为:

```
package fs.li4_7gridview;
import java.util.ArrayList;
import java.util.HashMap;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.GridView;
import android.widget.SimpleAdapter;
import android.widget.TextView;
```

```java
public class MainActivity extends Activity
{
    private TextView text = null;
    private int[] image = { R.drawable.po1, R.drawable.po2,
            R.drawable.po3, R.drawable.po5 };
    private String[] item = { "杀生丸", "宇智波佐助 ", "江户川柯南", "樱木花道" };
    private GridView gridView;
    private SimpleAdapter adapter;
    /** 第一次调用活动. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //通过 ID 查找到 main.xml 中的 TextView 控件
        text = (TextView) findViewById(R.id.text);
        //通过 ID 查找到 main.xml 中的 GridView 控件
        gridView = (GridView) findViewById(R.id.gridView);
        //创建一个 ArrayList 列表,内部存的是 HashMap 列表
        ArrayList<HashMap<String, Object>> listItems = new ArrayList<HashMap<String, Object
>>();
        //将数组信息分别存入 ArrayList 中
        int len = item.length;
        for(int i = 0; i < len ; i++){
            HashMap<String, Object> map = new HashMap<String, Object>();
            map.put("image", image[i]);
            map.put("item", item[i]);
            listItems.add(map);
        }
        //HashMap 中的 Key 信息要与 grid_item.xml 中的信息对应
        String[] from = {"image", "item"};
        //grid_item.xml 中对应的 ImageView 控件和 TextView 控件
        int[] to = {R.id.item_imageView, R.id.item_textView};
        //设定一个适配器
        adapter = new SimpleAdapter(this, listItems, R.layout.item, from, to);
        //对 GridView 进行适配
        gridView.setAdapter(adapter);
        //设置 GridView 的监听器
        gridView.setOnItemClickListener(new OnItemClickListener()
        {
            @Override
            public void onItemClick(AdapterView<?> arg0, View arg1,
                    int position, long arg3)
            {
                String str = "这次决斗" + item[position] + "赢了!";
                updateText(str);
            }
        });
    }
    private void updateText(String string)
    {
        //将文本信息设置给 TextView 控件并显示
        text.setText(string);
    }
}
```

运行程序，效果如图 4-7 所示。



图 4-7　GridView 视图

4.5　可扩展列表

1. ExpandableListView 控件概述

可展开的列表组件（ExpandableListView）为 ListView 的子类，在普通 ListView 的基础上进行扩展，把应用中的列表项分为几组，每组里又包含多个列表项。

ExpandableListView 的用法与普通 ListView 的用法非常类似，只是 ExpandableListView 显示的表项应该由 ExpandableListView 提供。表 4-4 列出了 ExpandableListView 额外支持的常用 XML 属性。

表 4-4　ExpandableListView 额外支持的常用 XML 属性

XML 属性	描　述
android:childDivider	指定各组内子列表项之间的分隔条
android:childIndicator	显示在子列表项旁边的 Drawable 对象
android:groupIndicator	显示在组列表项旁边的 Drawable 对象

2. ExpandableListView 控件经典案例

下面通过一个案例来演示 ExpandableListView 控件的用法。

【例 4-7】 利用 ExpandableListView 控件显示图片及相关信息。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_8ExpandableListView。

(2) 打开 res\layout 目录下的 main. xml 布局文件，在文件中声明一个 ExpandableListView 控件及两个 TextView 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
< TextView
    android:text = "动漫中各名人的能力"
    android:id = "@ + id/textView1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"/>
< ExpandableListView
    android:layout_height = "wrap_content"
    android:id = "@ + id/expandableListView"
    android:layout_width = "match_parent"/>
< TextView
    android:layout_height = "wrap_content"
    android:layout_width = "fill_parent"
    android:text = "@string/hello_world"
    android:id = "@ + id/text"/>
</LinearLayout >
```

(3) 打开 src\fs.li4_8expandablelistview 包下的 MainActivity.java 文件，在文件中实现当单击图片时，即显示图片中的相关信息，其代码为：

```
package fs.li4_8expandablelistview;
import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AbsListView;
import android.widget.BaseExpandableListAdapter;
import android.widget.ExpandableListAdapter;
import android.widget.ExpandableListView;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
public class MainActivity extends Activity
{
    private TextView text = null;
    private int[] image = { R.drawable.po1, R.drawable.po2,
            R.drawable.po3, R.drawable.po5 };
    private String[] item = { "杀生丸", "宇智波佐助 ", "江户川柯南", "樱木花道" };
    private String[][] ability = { { "会必杀技", "会斗鬼神" },
            { "会变身术", "会 分身术", "会替身术","会火遁凤仙火之术" }, { "会侦探", "会玩球
术" }, { "隐藏技术高" } };
    private ExpandableListView explandListView;
    /** 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //通过 ID 查找 main.xml 中的 TextView 控件
        text = (TextView) findViewById(R.id.text);
```

```
        //通过 ID 查找 main.xml 中的 ExpandableListView 控件
    explandListView = (ExpandableListView) findViewById(R.id.expandableListView);
        //设置 ExpandableListView 适配器
        ExpandableListAdapter adapter = new BaseExpandableListAdapter()
        {
            //处理子项目的单击事件
            @Override
            public boolean isChildSelectable(int groupPosition, int childPosition)
            {
                String str = item[groupPosition]
                        + ability[groupPosition][childPosition];
                updateText(str);
                return true;
            }
            @Override
            public boolean hasStableIds()
            {
                return true;
            }
            //返回父项目的视图控件
            @Override
            public View getGroupView(int groupPosition, boolean isExpanded,
                    View convertView, ViewGroup parent)
            {
                //新建一个线性布局
                LinearLayout ll = new LinearLayout(MainActivity.this);
                //设置布局样式为 Horizontal
                ll.setOrientation(0);
                //设置布局左边距为 50 像素
                ll.setPadding(50, 0, 0, 0);
                //新建一个 ImageView 对象
                ImageView imageView = new ImageView(MainActivity.this);
                //设置 ImageView 要显示的对象 ID
                imageView.setImageResource(image[groupPosition]);
                //将 ImageView 加到线性布局中
                ll.addView(imageView);
                //使用自定义文本框
                TextView textView = getTextView();
                //设置文本框里显示内容
                textView.setText(getGroup(groupPosition).toString());
                //将 TextView 加到线性布局中
                ll.addView(textView);
                return ll;
            }
            //返回父控件的 ID
            @Override
            public long getGroupId(int groupPosition)
            {
                return groupPosition;
            }
            //返回父控件的总数
            @Override
            public int getGroupCount()
            {
                return ability.length;
```

```
            }
            //取得父控件对象
            @Override
            public Object getGroup(int groupPosition)
            {
                return item[groupPosition];
            }
            //取得子控件的数量
            @Override
            public int getChildrenCount(int groupPosition)
            {
                return ability[groupPosition].length;
            }
            //取得子控件的视图
            @Override
            public View getChildView(int groupPosition, int childPosition,boolean isLastChild,
View convertView, ViewGroup parent)
            {
                //使用自定义 TextView 控件
                TextView textView = getTextView();
                //设置自定义 TextView 控件的内容
                textView.setText(getChild(groupPosition, childPosition).toString());
                return textView;
            }
            //取得子控件的 ID
            @Override
            public long getChildId(int groupPosition, int childPosition)
            {
                return childPosition;
            }
            //取得子控件的对象
            @Override
            public Object getChild(int groupPosition, int childPosition)
            {
                return ability[groupPosition][childPosition];
            }
           //自定义文本框
            public TextView getTextView()
            {
                AbsListView.LayoutParams lp = new AbsListView.LayoutParams(
                        ViewGroup.LayoutParams.FILL_PARENT, 64);
                TextView textView = new TextView(MainActivity.this);
                textView.setLayoutParams(lp);
                textView.setPadding(20, 0, 0, 0);
                //设置 TextView 控件为向左,水平居中对齐
                textView.setGravity(Gravity.CENTER_VERTICAL | Gravity.LEFT);
                return textView;
            }
        };
        explandListView.setAdapter(adapter);
    }
    private void updateText(String string)
    {
        //将文本信息设置给 TextView 控件显示出来
        text.setText(string);
```

```
    }
}
```

运行程序，默认效果如图 4-8 所示。当单击相关图片时，即显示相关的信息，如图 4-8(b)所示。



(a) 默认界面



(b) 浏览图像及相关信息

图 4-8　ExpandableListView 用法

4.6　网页浏览视图

网页浏览视图(WebView)类似于常用的浏览器，在 Android 手机中内置了一款高性能 WebKit 内核浏览器，WebView 组件就是由 WebKit 封装而来的，可以用来显示一个 Web 页面。

1. WebView 控件概述

WebView 是一个浏览器控件，通过这个控件可以直接访问网页，或者把输入的 HTML 字符串显示出来，功能比较强大，有以下几个优点：

- 功能强大，支持 CSS、JavaScript 等 HTML 语言，这样页面就能更漂亮；
- 能够对浏览器控件进行非常详细的设置，如字体大小、背景色、滚动条样式等；
- 能够捕捉到所有浏览器操作，如点击 URL，打开或关闭 URL；
- 能够很好地融入布局；
- 甚至 WebView 还能和 JavaScript 进行交互。

2. WebView 控件的经典案例

下面通过一个案例来演示 WebView 控件的用法。

【例 4-8】 本案例利用 WebView 控件实现打开对应的网页。其具体实现操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_9WebView。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 ScrollView 控件，在控件中定义一个 LinearLayout 布局及三个 WebView 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< ScrollView
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "wrap_content"
    android:orientation = "vertical"
    android:background = "#aabbcc">
    < LinearLayout
        android:orientation = "vertical"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:background = "#aabbcc">
        < WebView
            android:id = "@ + id/wv1"
            android:layout_height = "wrap_content"
            android:layout_width = "match_parent"/>
        < WebView
            android:id = "@ + id/wv2"
            android:layout_height = "wrap_content"
            android:layout_width = "match_parent"/>
        < WebView
            android:id = "@ + id/wv3"
            android:layout_height = "wrap_content"
            android:layout_width = "match_parent"/>
    </LinearLayout >
</ScrollView >
```

(3) 打开 src\fs.li4_9webview 包下的 MainActivity.java 文件,在文件中实现利用 WebView 打开对应的网页,其代码为:

```
package fs.li4_9webview;
import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);
        final String mimeType = "text/html";
        final String encoding = "utf - 8";
        WebView wv;
        wv = (WebView) findViewById(R.id.wv1);
        wv.loadDataWithBaseURL("http://www.google.com","< a href = 'http://www.baidu.com'>百度
搜索</a >", mimeType, encoding, "");
        wv = (WebView) findViewById(R.id.wv2);
        wv.loadDataWithBaseURL("http://www.google.com","< a href = 'www.cnblogs.com'>博客园</a >",
mimeType, encoding, "");
        //出现乱码,因此建议一般情况下不要使用此方法
         wv = (WebView) findViewById(R.id.wv3);
         wv.loadData("< a href = 'x'> http://ent.sina.com.cn/</a >", mimeType, encoding);
         }
}
```

运行程序,默认界面如图 4-9 所示。



图 4-9 WebView 的使用

4.7 图片切换器

1. ImageSwitcher 控件概述

ImageSwitcher 是一个图片切换控件，可以在一系列的图片中，逐张地显示特定的图片，利用该控件可以实现图片浏览器中的上一张、下一张的功能。其使用方法也较为简单，需要注意的是 ImageSwitcher 在使用时需要一个 ViewFactory，用来区分显示图片的容器和它的父窗口。

在 Android 中还有一个类似的组件就是 TextSwitcher，它们的用法基本相同。

使用 ImageSwitcher 或 TextSwitcher 必须设置一个 ViewFactory，用来在 ViewSwitcher 中创建 View，因此需要实现 ViewSwitcher. ViewFactory 接口，最后通过 makeView()方法创建相应的 View，即 ImageSwitcher 对应 ImageView，TextSwitcher 对应 TextView。

ImageSwitcher 的控件中的一些重要方法为：

- setImageURI(Uri uri)：设置图片地址。
- setImageResource(int resid)：设置图片资源库。
- setImageDrawable(Drawable drawable)：绘制图片。

2. ImageSwitcher 控件经典案例

下面通过一个案例来演示 ImageSwitcher 控件的用法。

【例 4-9】 使用 ImageSwitcher 控件实现图片的幻灯片效果。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_10ImageSwitcher。

(2) 打开 res\layout 目录下的 main. xml 布局文件，在文件中声明一个 ImageSwitcher 控件及 Gallery 控件，其代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
```

```
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc">
    < ImageSwitcher
        android:id = "@ + id/switcher"
        android:layout_width = "match_parent"
        android:layout_height = "match_parent"
        android:layout_alignParentTop = "true"
        android:layout_alignParentLeft = "true"/>
    < Gallery
        android:id = "@ + id/gallery"
        android:background = "#55000000"
        android:layout_width = "match_parent"
        android:layout_height = "60dp"
        android:layout_alignParentBottom = "true"
        android:layout_alignParentLeft = "true"
        android:gravity = "center_vertical"
        android:spacing = "16dp"/>
</RelativeLayout >
```

(3) 打开 src\fs.li4_10imageswitcher 包下的 MainActivity.java 文件，在文件中实现图片幻灯片效果，其代码为：

```
package fs.li4_10imageswitcher;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.view.animation.AnimationUtils;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageSwitcher;
import android.widget.ImageView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.Gallery.LayoutParams;
import android.widget.ViewSwitcher.ViewFactory;
public class MainActivity extends Activity implements
  OnItemSelectedListener, ViewFactory {
private ImageSwitcher is;
private Gallery gallery;
private Integer[] mThumbIds = { R.drawable.po1, R.drawable.po2,
   R.drawable.po3, R.drawable.po4, R.drawable.po5,};
@Override
protected void onCreate(Bundle savedInstanceState) {
  //TODO 自动存根法
  super.onCreate(savedInstanceState);
  requestWindowFeature(Window.FEATURE_NO_TITLE);
```

```
        setContentView(R.layout.main);
        is = (ImageSwitcher) findViewById(R.id.switcher);
        is.setFactory(this);
        is.setInAnimation(AnimationUtils.loadAnimation(this,
          android.R.anim.fade_in));
        is.setOutAnimation(AnimationUtils.loadAnimation(this,
          android.R.anim.fade_out));
        gallery = (Gallery) findViewById(R.id.gallery);
        gallery.setAdapter(new ImageAdapter(this));
        gallery.setOnItemSelectedListener(this);
    }
    @Override
    public View makeView() {
        ImageView i = new ImageView(this);
        i.setBackgroundColor(0xFF000000);
        i.setScaleType(ImageView.ScaleType.FIT_CENTER);
        i.setLayoutParams(new ImageSwitcher.LayoutParams(
          LayoutParams.MATCH_PARENT, LayoutParams.MATCH_PARENT));
        return i;
    }
    public class ImageAdapter extends BaseAdapter {
        public ImageAdapter(Context c) {
         mContext = c;
        }
        public int getCount() {
         return mThumbIds.length;
        }
        public Object getItem(int position) {
         return position;
        }
        public long getItemId(int position) {
         return position;
        }
        public View getView(int position, View convertView, ViewGroup parent) {
         ImageView i = new ImageView(mContext);
         i.setImageResource(mThumbIds[position]);
         i.setAdjustViewBounds(true);
         i.setLayoutParams(new Gallery.LayoutParams(
           LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
         i.setBackgroundResource(R.drawable.face);
         return i;
        }
        private Context mContext;
    }
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position,
         long id) {
        is.setImageResource(mThumbIds[position]);
    }
    @Override
    public void onNothingSelected(AdapterView<?> parent) {
        //TODO 自动存根法
    }
}
```

运行程序，效果如图 4-10 所示。



图 4-10　ImageSwitcher 实现图片幻灯效果

4.8　水平滚动视图

对于图片的显示我们还经常使用 HorizontalScrollView 控件或 ViewPager 控件。本节介绍 HorizontalScrollView 控件的使用。

1．HorizontalScrollView 控件概述

HorizontalScrollView 和 Gallery 一样是一种水平滚动视图，是用于布局的容器，可以放置让用户使用滚动条查看的视图层次结构，允许视图结构比手机的屏幕大。

HorizontalScrollView 继承至 FrameLayout，继承关系如图 4-11 所示。它是一种布局方式，其子项滚动查看时是整体移动的，并且子项本身可以是有复杂层次结构的布局管理器。常见的应用是子项在水平方向中，用户可以滚动显示顶层水平排列的子项(items)。

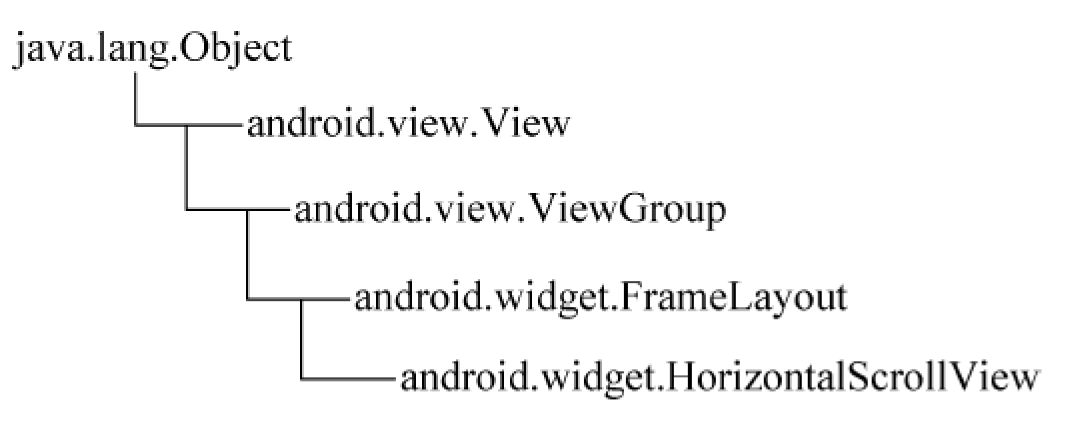


图 4-11　HorizontalScrollView 的继承树

HorizontalScrollView 只支持水平方向的滚动显示。

2．HorizontalScrollView 控件经典案例

下面通过一个案例来演示 HorizontalScrollView 控件的用法。

【例 4-10】　利用 HorizontalScrollView 控件实现图片的水平滚动。其实现操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_11HorizontalScrollView。

(2) 打开 res\layout 目录下的 main. xml 文件，在文件中定义一个 TextView 控件、HorizontalScrollView 及 LinearLayout 控件，其代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
```

```
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc">
    <TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_centerHorizontal = "true"
        android:layout_centerVertical = "true"
        android:text = "图片水平滚动" />
    <HorizontalScrollView
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content" >
        <LinearLayout
            android:orientation = "horizontal"
            android:id = "@ + id/myhorizon"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"/>
    </HorizontalScrollView>
</RelativeLayout>
```

(3) 打开 src\fs.li4_11horizontalscrollview 包下的 MainActivity.java 文件,在文件中实现图片的水平滚动,其代码为:

```
package fs.li4_11horizontalscrollview;
import android.os.Bundle;
import android.app.Activity;
import android.view.Gravity;
import android.view.Menu;
import android.view.View;
import android.view.ViewGroup.LayoutParams;
import android.widget.ImageView;
import android.widget.LinearLayout;
public class MainActivity extends Activity {
private LinearLayout myhorizonLayout;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        myhorizonLayout = (LinearLayout) findViewById(R.id.myhorizon);
        int[] imageIDs = {
                R.drawable.g1,R.drawable.g2,R.drawable.g3,
                R.drawable.g4,R.drawable.g5
        };
        for(Integer id:imageIDs){
            myhorizonLayout.addView(insertImage(id));
        }
    }
    private View insertImage(Integer id) {
        LinearLayout layout = new LinearLayout(getApplicationContext());
        layout.setLayoutParams(new LayoutParams(320,320));
        layout.setGravity(Gravity.CENTER);
        ImageView imageView = new ImageView(getApplicationContext());
        imageView.setLayoutParams(new LayoutParams(300,300));
        imageView.setBackgroundResource(id);
```

```
            layout.addView(imageView);
            return layout;
        }
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            getMenuInflater().inflate(R.menu.main, menu);
            return true;
        }
}
```

运行程序,效果如图 4-12 所示,当用鼠标水平拖曳图片,可浏览图像。



图 4-12　图片的水平滚动

4.9　多 页 视 图

本节将介绍 ViewPage 控件来实现图片的浏览效果。

1. ViewPage 控件概述

Android 的左右滑动在实际编程经常能用到,如查看多张图片,左右切换 Tab 页。自 Android 3.0 之后的 SDK 中提供了 android-support-v4 包用以实现版本兼容,让老版本系统下的应用通过加入 jar 包实现扩展,其中有一个可以实现左右滑动的类 ViewPager。

2. ViewPage 控件经典案例

下面通过一个案例来演示 ViewPage 控件的用法。

【例 4-11】 利用 ViewPage 控件浏览图片。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li4_12ViewPage。

(2) 打开 res\layout 目录下的 main.xml 布局文件,在文件中定义一个 ViewPager 控件及一个 PagerTitleStrip 控件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
```

```
    android:orientation = "vertical"
    android:background = "#aabbcc">
<android.support.v4.view.ViewPager
        android:id = "@+id/viewpager"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_gravity = "center" >
<android.support.v4.view.PagerTitleStrip
        android:id = "@+id/pagertitle"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_gravity = "top" />
    </android.support.v4.view.ViewPager>
</LinearLayout>
```

（3）打开 src\fs.li4_12viewpage 包下的 MainActivity.java 文件，用于实现图像的翻页浏览，其代码为：

```
package fs.li4_12viewpage;
import java.util.ArrayList;
import android.os.Bundle;
import android.app.Activity;
import android.graphics.drawable.Drawable;
import android.support.v4.view.PagerAdapter;
import android.support.v4.view.PagerTitleStrip;
import android.support.v4.view.ViewPager;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
public class MainActivity extends Activity {
    /** 第一次调用活动. */
    private ViewPager mViewPager;
    private PagerTitleStrip mPagerTitleStrip;
    private int[] pics = { R.drawable.g1, R.drawable.g2, R.drawable.g4 };
    final ArrayList<View> views = new ArrayList<View>();
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mViewPager = (ViewPager) findViewById(R.id.viewpager);
        mPagerTitleStrip = (PagerTitleStrip) findViewById(R.id.pagertitle);
        LinearLayout.LayoutParams mParams = new LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.WRAP_CONTENT,
                LinearLayout.LayoutParams.WRAP_CONTENT);
        //将要分页显示的 View 装入数组中
        for (int i = 0; i < pics.length; i++) {
            ImageView iv = new ImageView(this);
            iv.setLayoutParams(mParams);
            iv.setImageResource(pics[i]);
            views.add(iv);
        }
        //每个页面的 Title 数据
        final ArrayList<String> titles = new ArrayList<String>();
        titles.add("tab1");
        titles.add("tab2");
        titles.add("tab3");
```

```
        //填充 ViewPager 的数据适配器
        PagerAdapter mPagerAdapter = new PagerAdapter() {
            @Override
            public boolean isViewFromObject(View arg0, Object arg1) {
                return arg0 == arg1;
            }
            @Override
            public int getCount() {
                return views.size();
            }
            @Override
            public void destroyItem(View container, int position, Object object) {
                ((ViewPager) container).removeView(views.get(position));
            }
            @Override
            public CharSequence getPageTitle(int position) {
                return titles.get(position);
            }
            @Override
            public Object instantiateItem(View container, int position) {
                ((ViewPager) container).addView(views.get(position));
                return views.get(position);
            }
        };
        mViewPager.setAdapter(mPagerAdapter);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

运行程序,效果如图 4-13 所示。



图 4-13　图片的翻页浏览

4.10 滑动式抽屉

1. SlidingDrawer 控件概述

SlidingDrawer 隐藏屏外的内容，并允许用户通过手柄来显示隐藏内容。它可以垂直或水平滑动，它由两方面内容组成，其一是可以拖动的手柄；其二是隐藏的内容。它里面的控件必须设置布局，在布局文件中必须指定手柄和内容。

其重要的属性如下：

- android:allowSingleTap：指示是否可以通过 handle 打开或关闭。
- android:animateOnClick：指示是否当使用者按下手柄打开/关闭时是否该有一个动画。
- android:content：隐藏的内容。
- android:handle：手柄。

其常用的重要方法主要有：

- animateClose()：关闭时实现动画。
- close()：即时关闭。
- getContent()：获取内容。
- isMoving()：指示 SlidingDrawer 是否在移动。
- isOpened()：指示 SlidingDrawer 是否已全部打开。
- lock()：屏蔽触摸事件。
- setOnDrawerCloseListener(SlidingDrawer. OnDrawerCloseListener onDrawerCloseListener)：SlidingDrawer 关闭时调用。
- unlock()：解除屏蔽触摸事件。
- toggle()：切换打开和关闭的抽屉 SlidingDrawer。

2. SlidingDrawer 控件经典案例

下面通过一个案例来演示 SlidingDrawer 控件的用法。

【例 4-12】 利用 SlidingDrawer 控件实现图像的拖动。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_13SlidingDrawer。

(2) 打开 res\layout 目录下的 main. xml 布局文件，在文件中声明一个 SlidingDrawer 控件、两个 LinearLayout 布局及一个 TextView 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
      android:layout_width = "fill_parent"
      android:layout_height = "fill_parent"
      android:orientation = "vertical"
      android:background = "#aabbcc">
<SlidingDrawer
      android:layout_width = "fill_parent"
      android:layout_height = "fill_parent"
      android:handle = "@ + id/handle"
      android:content = "@ + id/content"
      android:orientation = "vertical"
      android:id = "@ + id/slidingdrawer">
```

```
<ImageButton
    android:id="@id/handle"
    android:layout_width="50dip"
    android:layout_height="44dip"
    android:src="@drawable/b4" />
<LinearLayout
    android:id="@id/content"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffffff">
 <TextView
     android:text="这是一个滑动式抽屉的示例"
     android:id="@+id/tv"
     android:textSize="18px"
     android:textColor="#000000"
     android:gravity="center_vertical|center_horizontal"
     android:layout_width="match_parent"
     android:textStyle="bold"
     android:layout_height="match_parent"/>
</LinearLayout>
</SlidingDrawer>
</LinearLayout>
```

(3) 打开 src\fs.li4_13slidingdrawer 包下的 MainActivity.java 文件，在文件中实现图片的拖动及变换，其代码为：

```
package fs.li4_13slidingdrawer;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ImageButton;
import android.widget.SlidingDrawer;
import android.widget.TextView;
public class MainActivity extends Activity {
    private SlidingDrawer mDrawer;
    private ImageButton imbg;
    private Boolean flag=false;
    private TextView tv;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //TODO 自动存根法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        imbg=(ImageButton)findViewById(R.id.handle);
        mDrawer=(SlidingDrawer)findViewById(R.id.slidingdrawer);
        tv=(TextView)findViewById(R.id.tv);
        mDrawer.setOnDrawerOpenListener(new SlidingDrawer.OnDrawerOpenListener(){
            @Override
            public void onDrawerOpened() {
                flag=true;
                imbg.setImageResource(R.drawable.b4);
            }
        });
        mDrawer.setOnDrawerCloseListener(new SlidingDrawer.OnDrawerCloseListener(){
            @Override
            public void onDrawerClosed() {
                flag=false;
```

```
                    imbg.setImageResource(R.drawable.b6);
                    }
            });
            mDrawer.setOnDrawerScrollListener(new SlidingDrawer.OnDrawerScrollListener(){
                @Override
                public void onScrollEnded() {
                    tv.setText("结束拖动");
                    }
                @Override
                public void onScrollStarted() {
                    tv.setText("开始拖动");
                    }
            });
        }
    }
```

运行程序，默认效果如图 4-14(a)所示；把图 4-14(a)中图片向上拖动，效果如图 4-14(b)所示；把图 4-14(b)中的图片向下拖动，效果如图 4-14(c)所示，这时图片就切换了。



(a) 默认效果　　(b) 图像向上拖动　　(c) 图像向下拖动并切换图像

图 4-14　滑动式抽屉

4.11　画 廊 视 图

画廊视图是一种较为常见的控件，其效果酷炫，且使用方式简单，是设计相册或图片选择器时首选的控件。

1. Gallery 控件概述

Gallery 是一种水平滚动的列表，一般情况下用来显示图片等资源，可以使图片在屏幕上滑动。Gallery 所显示的图片资源同样来自于适配器。

Gallery 是 View 的子类，它的继承树如图 4-15 所示。

```
Java.lang.Object
  └─Android.view.View
      └─Android.view.ViewGroup
          └─Android.widget.AdapterView
              └─Android.widget.AbsSpinner
                  └─Android.widget.Gallery
```

图 4-15　Gallery 的继承树

Gallery 额外提供了如表 4-5 所示的 XML 属性。

表 4-5　Gallery 常用的 XML 属性及描述

XML 属性	方　　法	描　　述
android:animationDuration	setAnimationDuration(int)	设置列表项切换时的动画持续时间
android:gravity	setGravity(int)	设置对齐方式
android:spacing	setSpacing(int)	设置 Gallery 内列表项之间的间距
android:unselectedAlpah	setUnselectedAlpha(float)	设置没有选中的列表项的透明度

Gallery 本身的用法非常简单——基本上与 Spinner 的用法形似，只要为它提供一个内容 Adapter 即可，该 Adapter 的 getView 方法所返回的 View 将作为 Gallery 列表的列表项；如果程序需要监控 Gallery 选择项的改变，可以通过为 Gallery 添加 onItemSelectedListener 监听器实现。

2. Gallery 控件经典案例

下面通过一个案例来演示 Gallery 控件的用法。

【例 4-13】 使用 Gallery 控件实现图片的浏览。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li4_14Gallery。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 ImageSwitcher 控件及一个 Gallery 控件，其代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc">
   <ImageSwitcher
       android:id = "@ + id/switcher"
       android:layout_width = "fill_parent"
       android:layout_height = "fill_parent"
       android:layout_alignParentTop = "true"
       android:layout_alignParentLeft = "true"/>
  <Gallery
      android:id = "@ + id/gallery"
      android:background = "#55000000"
      android:layout_width = "fill_parent"
      android:layout_height = "60dp"
      android:layout_alignParentBottom = "true"
      android:layout_alignParentLeft = "true"
      android:gravity = "center_vertical"
      android:spacing = "16dp"/>
</RelativeLayout>
```

(3) 打开 src\fs.li4_14gallery 包下的 MainActivity.java 文件，在文件中实现图片的浏览，其代码为：

```
package fs.li4_14gallery;
```

```
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.view.animation.AnimationUtils;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageSwitcher;
import android.widget.ImageView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.Gallery.LayoutParams;
import android.widget.ViewSwitcher.ViewFactory;
public class MainActivity extends Activity implements OnItemSelectedListener,ViewFactory {
    private ImageSwitcher mSwitcher;
    private Integer[] mThumbIds = { R.drawable.c1,R.drawable.c2, R.drawable.c3,R.drawable.c4,
R.drawable.c5,R.drawable.g1, R.drawable.c7,R.drawable.c8 };
    private Integer[] mImageIds = { R.drawable.d1, R.drawable.d2,R.drawable.d3, R.drawable.d4,
R.drawable.d5,R.drawable.d6, R.drawable.d7, R.drawable.d8 };
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.main);
        mSwitcher = (ImageSwitcher) findViewById(R.id.switcher);
        mSwitcher.setFactory(this);mSwitcher.setInAnimation(AnimationUtils.loadAnimation
(this,android.R.anim.fade_in));mSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
android.R.anim.fade_out));
        Gallery g = (Gallery) findViewById(R.id.gallery);
        g.setAdapter(new ImageAdapter(this));
        g.setOnItemSelectedListener(this);
    }
    public class ImageAdapter extends BaseAdapter {
        public ImageAdapter(Context c) {
            mContext = c;
        }
        public int getCount() {
            return mThumbIds.length;
        }
        public Object getItem(int position) {
            return position;
        }
        public long getItemId(int position) {
            return position;
        }
        public View getView(int position, View convertView, ViewGroup parent) {
            ImageView i = new ImageView(mContext);
            i.setImageResource(mThumbIds[position]);
            i.setAdjustViewBounds(true);
            i.setLayoutParams(new Gallery.LayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.
WRAP_CONTENT));
            i.setBackgroundResource(R.drawable.dcuk2);
            return i;
```

```
        }
        private Context mContext;
    }
    @Override
    public void onItemSelected(AdapterView<?> adapter, View v, int position,long id) {
        mSwitcher.setImageResource(mImageIds[position]);
    }
    @Override
    public void onNothingSelected(AdapterView<?> arg0) {}
    @Override
    public View makeView() {
        ImageView i = new ImageView(this);
        i.setBackgroundColor(0xFF000000);
        i.setScaleType(ImageView.ScaleType.FIT_CENTER);
        i.setLayoutParams(new ImageSwitcher.LayoutParams(LayoutParams.FILL_PARENT, LayoutParams.
FILL_PARENT));
        return i;
    }
}
```

运行程序,效果如图 4-16 所示。



图 4-16　Gallery 控件的用法

4.12　2D 视图

1. graphics 类

在 Android 中需要通过 graphics 类来显示 2D 图形。graphics 类中包括了 Canvas(画布)、Paint(画笔)、Color(颜色)、Bitmap(图像)等常用的类。graphics 具有绘制点、线、颜色、2D 几何图形、图像处理等功能。

1) Color(颜色)类

Android 系统中颜色的常用表示方法有以下三种:

(1) int color=Color.BLUE;

(2) int color=Color.argb(150,200,0,100);

(3) 在 xml 文件中定义颜色。

在实际应用当中,常用的颜色有以下一些,其颜色常量及其表示的颜色如下所示:

Color.BLACK:黑色

Color.GREEN:绿色

Color.BLUE:蓝色

Color.LTGRAY:浅灰色

Color.CYAN:青绿色

Color.MAGENTA:红紫色

Color.DKGRAY:灰黑色

Color.RED:红色

Color.YELLOW:黄色

Color.TRANSPARENT:透明

Color.GRAY:灰色

Color.WHITE:白色

2) Paint(画笔)类

要绘制图形,首先得调整画笔,按照自己的开发需要设置画笔的相关属性。Pain 类的常用属性设置方法如下:

(1) 图形绘制。

Paint 类实现图形绘制的主要方法如下:

- setARGB(int a,int r,int g,int b):设置绘制的颜色,a 代表透明度,r、g、b 代表颜色值。
- setAlpha(int a):设置绘制图形的透明度。
- setColor(int color):设置绘制的颜色,使用颜色值来表示,该颜色值包括透明度和 RGB 颜色。
- setAntiAlias(boolean aa):设置是否使用抗锯齿功能,会消耗较大资源,绘制图形速度会变慢。
- setDither(boolean dither):设定是否使用图像抖动处理,会使绘制的图片颜色更加平滑和饱满,图像更加清晰。
- setFilterBitmap(boolean filter):如果该项设置为 true,则图像在动画进行中会滤掉对 Bitmap 图像的优化操作,加快显示速度,本设置项依赖于 dither 和 xfermode 的设置。
- setMaskFilter(MaskFilter maskfilter):设置 MaskFilter,可以用不同的 MaskFilter 实现滤镜的效果,如滤化,立体等。
- setColorFilter(ColorFilter colorfilter):设置颜色过滤器,可以在绘制颜色时实现不同颜色的变换效果。
- setPathEffect(PathEffect effect):设置绘制路径的效果,如点画线等。
- setShader(Shader shader):设置图像效果,使用 Shader 可以绘制各种渐变效果。
- setShadowLayer(float radius,float dx,float dy,int color):在图形下面设置阴影层,产生阴影效果,radius 为阴影的角度,dx 和 dy 为阴影在 x 轴和 y 轴上的距离,color 为

阴影的颜色。

- setStyle(Paint. Style style)：设置画笔的样式，取值为 FILL、FILL_OR_STROKE 或 STROKE。
- setStrokeCap(Paint. Cap cap)：当画笔样式为 STROKE 或 FILL_OR_STROKE 时，设置笔刷的图形样式，如圆形样式(Cap. ROUND)或方形样式(Cap. SQUARE)。
- setSrokeJoin(Paint. Join join)：设置绘制时各图形的结合方式，如平滑效果等。
- setStrokeWidth(float width)：当画笔样式为 STROKE 或 FILL_OR_STROKE 时，设置笔刷的粗细度。
- setXfermode(Xfermode xfermode)：设置图形重叠时的处理方式，如合并、交集或并集，经常用来制作橡皮的擦除效果。

(2) 文本绘制。

Paint 类实现文本绘制的主要方法如下：

- setFakeBoldText(boolean fakeBoldText)：模拟实现粗体文字，设置在小字体上效果会非常差。
- setSubpixelText(boolean subpixelText)：设置该项为 true，将有助于文本在 LCD 屏幕上的显示效果。
- setTextAlign(Paint. Align align)：设置绘制文字的对齐方向。
- setTextScaleX(float scaleX)：设置绘制文字 x 轴的缩放比例，可以实现文字的拉伸的效果。
- setTextSize(float textSize)：设置绘制文字的字号大小。
- setTextSkewX(float skewX)：设置斜体文字，skewX 为倾斜弧度。
- setTypeface(Typeface typeface)：设置 Typeface 对象，即字体风格，包括粗体，斜体以及衬线体，非衬线体等。
- setUnderlineText(boolean underlineText)：设置带有下划线的文字效果。
- setStrikeThruText(boolean strikeThruText)：设置带有删除线的效果。

3) Canvas(画布)类

画笔属性设置好之后，还需要将图像绘制到画布上。Canvas 类可以用来实现各种图形的绘制工作，如绘制直线、矩形、圆等。Canvas 绘制常用图形的方法如下：

绘制直线：canvas. drawLine(float startX, float startY, float stopX, float stopY, Paint paint);

绘制矩形：canvas. drawRect(float left, float top, float right, float bottom, Paint paint);

绘制圆形：canvas. drawCircle(float cx, float cy, float radius, Paint paint);

绘制字符：canvas. drawText(String text, float x, float y, Paint paint);

绘制图形：canvas. drawBirmap(Bitmap bitmap, float left, float top, Paint paint);

4) 自定义 View 的基本实现方法

首先，需要自定义一个类，如 MyView，继承 View 类；然后，重写 View 类的 onDraw()函数；最后，在 onDraw()函数中使用 Paint 和 Canvas 对象绘制需要的图形。

2. 经典案例

下面通过一个经典案例来演示使用 graphics 类来绘制了一幅简单的北京奥运宣传画，包

括奥运五环、“北京欢迎您”的宣传标语以及图片。其具体实现步骤为：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 Paint_Canvas_Test。

（2）res\layout 目录下的布局文件 main.xml 采用默认代码。

（3）在 src\paint_canvas_test 包下新建一个自定义一个类 Custom_View，在类中重写了 onDraw()函数，并定义几种不同的画笔，分别用来绘制各种颜色的奥运五环以及绘制字符串“北京欢迎您”等，其代码为：

```
package com.paint_canvas_test;
import android.view.View;
import android.content.Context;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Paint.Style;
public class Custom_View extends View {
public Custom_View(Context context) {
    super(context);
    }
public void onDraw(Canvas canvas) {
    Paint paint_blue = new Paint();                     //绘制蓝色的环
    paint_blue.setColor(Color.BLUE);
    paint_blue.setStyle(Style.STROKE);
    paint_blue.setStrokeWidth(10);
    canvas.drawCircle(110,150,60,paint_blue);
    Paint paint_yellow = new Paint();                   //绘制黄色的环
    paint_yellow.setColor(Color.YELLOW);
    paint_yellow.setStyle(Style.STROKE);
    paint_yellow.setStrokeWidth(10);
    canvas.drawCircle((float)175.5, 210, 60, paint_yellow);
    Paint paint_black = new Paint();                    //绘制黑色的环
    paint_black.setColor(Color.BLACK);
    paint_black.setStyle(Style.STROKE);
    paint_black.setStrokeWidth(10);
    canvas.drawCircle(245, 150, 60, paint_black);
    Paint paint_green = new Paint();                    //绘制绿色的环
    paint_green.setColor(Color.GREEN);
    paint_green.setStyle(Style.STROKE);
    paint_green.setStrokeWidth(10);
    canvas.drawCircle(311, 210, 60, paint_green);
    Paint paint_red = new Paint();                      //绘制红色的环
    paint_red.setColor(Color.RED);
    paint_red.setStyle(Style.STROKE);
    paint_red.setStrokeWidth(10);
    canvas.drawCircle(380, 150, 60, paint_red);
    Paint paint_string = new Paint();                   //绘制字符串
    paint_string.setColor(Color.BLUE);
    paint_string.setTextSize(20);
    canvas.drawText("Welcome to Beijing", 245, 310, paint_string);
    Paint paint_line = new Paint();                     //绘制直线
    paint_line.setColor(Color.BLUE);
    canvas.drawLine(240, 310, 425, 310, paint_line);
    Paint paint_text = new Paint();                     //绘制字符串
    paint_text.setColor(Color.BLUE);
```

```
        paint_text.setTextSize(20);
        canvas.drawText("北京欢迎您", 275, 330, paint_text);
        //载入图片
        canvas.drawBitmap(BitmapFactory.decodeResource(getResources(),R.drawable.kj), 35, 340,
    paint_line);
        }
    }
```

（4）打开 src\paint_canvas_test 包下的 MainActivity 文件，在文件中实现将自定义的 MyView 视图显示到手机屏幕上，即加载 MyView 视图，可以使用 setContentView()方法，其代码为：

```
package com.paint_canvas_test;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.MenuItem;
import android.support.v4.app.NavUtils;
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(newCustom_View(this));      //加载 MyView
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
       return true;
    }
}
```

运行程序，效果如图 4-17 所示。



图 4-17　绘制 2D 图形效果

4.13 Path 菜单

1. Path 类概述

Android 提供的 Path 为一个非常实用的类，可预先在 View 上将 N 个点连成一条“路径”，然后调用 Canvas 的 drawPath(path，paint)即可沿着路径绘制图形。实际上 Android 还为路径绘制提供了 PathEffect 来定义绘制效果，PathEffect 包含了如下子类(每个子类代表一种绘制效果)：

- ComposePathEffect；
- CornerPathEffect；
- DashPathEffect；
- DiscretePathEffect；
- PathDashPathEffect；
- SumPathEffect。

2. 经典案例

下面通过一个经典案例来演示 Path 类的用法，实现超仿 Path 菜单。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Paint_2DTest。

(2) 在 res\value 目录新建一个 attrs. xml 文件，用于设置 Android 图标按钮的位置，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < declare - styleable name = "PathMenuView">
        < attr name = "position">
            < enum name = "left_top" value = "0"></enum >
            < enum name = "right_top" value = "1"></enum >
            < enum name = "right_bottom" value = "2"></enum >
            < enum name = "left_bottom" value = "3"></enum >
        </attr >
    </declare - styleable >
</resources >
```

(3) 在 src\paint_2dtest 包下新建一个 PathMenuView. java，这个就是自定义的 Path 菜单控件，其代码为：

```
package fs.paint_2dtest;
import android.content.Context;
import android.content.res.TypedArray;
import android.util.AttributeSet;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.Animation;
import android.view.animation.AnticipateInterpolator;
import android.view.animation.OvershootInterpolator;
import android.view.animation.TranslateAnimation;
import android.widget.FrameLayout;
import android.widget.ImageView;
/**
```

```
 * 仿 path 菜单
 * position 定义菜单的位置,目前支持左上、右上、右下、左下 4 个方向
 * menuResIds 定义出现的菜单的资源 ID
 */
public class PathMenuView extends FrameLayout {
    private static final int LEFT_TOP = 0;
    private static final int RIGHT_TOP = 1;
    private static final int RIGHT_BOTTOM = 2;
    private static final int LEFT_BOTTOM = 3;
    /**
     * 默认的位置是在右下角
     */
    private int position = 3;
    /**
     * 那个圆形菜单
     */
    private ImageView mHome;
    /**
     * 上下文
     */
    private Context mContext;
    /**
     * 设备的宽度
     */
    private int mWIDTH = 0;
    /**
     * 设备的高度
     */
    private int mHEIGHT = 0;
    /**
     * 设备的 density
     */
    private float mDensity;
    /**
     * 菜单是否显示
     */
    private boolean bMenuShow;
    private static int xOffset = 15;
    private static int yOffset = -13;
    /**
     * 菜单的资源个数
     */
    private int[] menuResIds = {R.drawable.p1, R.drawable.p2, R.drawable.p3, R.drawable.p4,
R.drawable.p5};
    public PathMenuView(Context context){
        super(context);
        setupViews();
    }
    public PathMenuView(Context context, AttributeSet attrs) {
        super(context, attrs);
        TypedArray a = context.obtainStyledAttributes(attrs,
                R.styleable.PathMenuView);
        position = a.getInt(R.styleable.PathMenuView_position,3);
        a.recycle();
        setupViews();
```

```
    }
    private void setupViews(){
        mContext = getContext();
        mHEIGHT = mContext.getResources().getDisplayMetrics().heightPixels;
        mWIDTH = mContext.getResources().getDisplayMetrics().widthPixels;
        mDensity = mContext.getResources().getDisplayMetrics().density;
        xOffset = (int) (10.667 * mDensity);
        yOffset = (int) (8.667 * mDensity);
        mHome = new ImageView(mContext);
        mHome.setImageResource(R.drawable.ic_launcher);
        mHome.setOnClickListener(listener);
        addView(mHome);
        LayoutParams mHomeparams = (FrameLayout.LayoutParams)mHome.getLayoutParams();
        mHomeparams.width = LayoutParams.WRAP_CONTENT;
        mHomeparams.height = LayoutParams.WRAP_CONTENT;
        switch (position) {
        case LEFT_TOP:
            mHomeparams.gravity = Gravity.LEFT | Gravity.TOP;
            for (int i = 0; i < menuResIds.length; i++) {
                int width_padding = mWIDTH / ((menuResIds.length - 1) * 2);
                int height_padding = mHEIGHT / ((menuResIds.length - 1) * 2);
                ImageView imageView = new ImageView(mContext);
                imageView.setImageResource(menuResIds[i]);
                addView(imageView);
                LayoutParams params = (FrameLayout.LayoutParams) imageView
                        .getLayoutParams();
                params.width = LayoutParams.WRAP_CONTENT;
                params.height = LayoutParams.WRAP_CONTENT;
                params.leftMargin = mWIDTH/2
                        - ((menuResIds.length - i - 1) * width_padding);
                params.topMargin = mHEIGHT/2 - i * height_padding;
                params.gravity = Gravity.LEFT|Gravity.TOP;
                imageView.setLayoutParams(params);
            }
            break;
        case RIGHT_TOP:
            mHomeparams.gravity = Gravity.RIGHT | Gravity.TOP;
            for (int i = 0; i < menuResIds.length; i++) {
                int width_padding = mWIDTH / ((menuResIds.length - 1) * 2);
                int height_padding = mHEIGHT / ((menuResIds.length - 1) * 2);
                ImageView imageView = new ImageView(mContext);
                imageView.setImageResource(menuResIds[i]);
                addView(imageView);
                LayoutParams params = (FrameLayout.LayoutParams) imageView
                        .getLayoutParams();
                params.width = LayoutParams.WRAP_CONTENT;
                params.height = LayoutParams.WRAP_CONTENT;
                params.rightMargin = mWIDTH/2
                        - ((menuResIds.length - i - 1) * width_padding);
                params.topMargin = mHEIGHT/2 - i * height_padding;
                params.gravity = Gravity.RIGHT|Gravity.TOP;
                imageView.setLayoutParams(params);
            }
            break;
        case RIGHT_BOTTOM:
```

```
            mHomeparams.gravity = Gravity.RIGHT | Gravity.BOTTOM;
            for (int i = 0; i < menuResIds.length; i++) {
                int width_padding = mWIDTH / ((menuResIds.length - 1) * 2);
                int height_padding = mHEIGHT / ((menuResIds.length - 1) * 2);
                ImageView imageView = new ImageView(mContext);
                imageView.setImageResource(menuResIds[i]);
                addView(imageView);
                LayoutParams params = (FrameLayout.LayoutParams) imageView
                        .getLayoutParams();
                params.width = LayoutParams.WRAP_CONTENT;
                params.height = LayoutParams.WRAP_CONTENT;
                params.rightMargin = mWIDTH / 2
                        - ((menuResIds.length - i - 1) * width_padding);
                params.bottomMargin = mHEIGHT / 2 - i * height_padding;
                params.gravity = Gravity.RIGHT | Gravity.BOTTOM;
                imageView.setLayoutParams(params);
            }
            break;
        case LEFT_BOTTOM:
            mHomeparams.gravity = Gravity.LEFT | Gravity.BOTTOM;
            for(int i = 0; i < menuResIds.length; i++){
                int width_padding = mWIDTH / ((menuResIds.length - 1) * 2);
                int height_padding = mHEIGHT / ((menuResIds.length  - 1) * 2);
                ImageView imageView = new ImageView(mContext);
                imageView.setImageResource(menuResIds[i]);
                addView(imageView);
                LayoutParams params = (FrameLayout.LayoutParams)imageView.getLayoutParams();
                params.width = LayoutParams.WRAP_CONTENT;
                params.height = LayoutParams.WRAP_CONTENT;
    params.leftMargin = mWIDTH / 2 - ((menuResIds.length - i - 1) * width_padding);
                params.bottomMargin = mHEIGHT / 2 - i * height_padding;
                params.gravity = Gravity.LEFT | Gravity.BOTTOM;
                imageView.setLayoutParams(params);
            }
            break;
        default:
                break;
        }
        mHome.setLayoutParams(mHomeparams);
    }
    private OnClickListener listener = new OnClickListener() {
        public void onClick(View v) {
            if (!bMenuShow) {
                startAnimationIn(PathMenuView.this, 300);
            } else {
                startAnimationOut(PathMenuView.this, 300);
            }
            bMenuShow = !bMenuShow;
        }
    };
    /**
      * 菜单隐藏动画
      */
    private void startAnimationIn(ViewGroup group, int duration) {
        for (int i = 1; i < group.getChildCount(); i++) {
            ImageView imageview = (ImageView) group.getChildAt(i);
```

```
                imageview.setVisibility(0);
                MarginLayoutParams mlp = (MarginLayoutParams) imageview
                        .getLayoutParams();
                Animation animation = null;
                switch (position) {
                case LEFT_TOP:
                    animation = new TranslateAnimation(0F, - mlp.leftMargin + xOffset,0F, - mlp
.topMargin + yOffset);
                    break;
                case RIGHT_TOP:
                    animation = new TranslateAnimation(mlp.rightMargin - xOffset,0F, - mlp
.topMargin + yOffset,0F);
                    break;
                case LEFT_BOTTOM:
                    animation = new TranslateAnimation(0F,  - mlp.leftMargin + xOffset, 0F,
- yOffset + mlp.bottomMargin);
                    break;

                case RIGHT_BOTTOM:
                    animation = new TranslateAnimation(mlp.rightMargin - xOffset,0F, - yOffset
+ mlp.bottomMargin, 0F);
                    break;
                default:
                    break;
                }
                animation.setFillAfter(true);
                animation.setDuration(duration);
                animation.setStartOffset((i * 100) / ( - 1 + group.getChildCount()));
                animation.setInterpolator(new OvershootInterpolator(2F));
                imageview.startAnimation(animation);
            }
        }
        /**
          * 菜单显示动画
          */
        private void startAnimationOut(ViewGroup group,int duration){
            for (int i = 1; i < group.getChildCount(); i++) {
                final ImageView imageview = (ImageView) group
                        .getChildAt(i);
        MarginLayoutParams mlp = (MarginLayoutParams) imageview.getLayoutParams();
                Animation animation = null;
                switch (position) {
                case LEFT_TOP:
                    animation = new TranslateAnimation( - mlp.leftMargin + xOffset,0F, - mlp
.topMargin + yOffset,0F);
                    break;
                case RIGHT_TOP:
                    animation = new TranslateAnimation(0F,mlp.rightMargin - xOffset,0F, - mlp
.topMargin + yOffset);
                    break;
                case LEFT_BOTTOM:
                    animation = new TranslateAnimation( - mlp.leftMargin + xOffset,0F,  - yOffset
+ mlp.bottomMargin,0F);
                    break;
                case RIGHT_BOTTOM:
                    animation = new TranslateAnimation(0F,mlp.rightMargin - xOffset, 0F, - yOffset
+ mlp.bottomMargin);
                    break;
```

```
                default:
                    break;
                }
                animation.setFillAfter(true);animation.setDuration(duration);
                animation.setStartOffset(((group.getChildCount() - i) * 100)/ ( - 1 + group
.getChildCount()));
                animation.setInterpolator(new AnticipateInterpolator(2F));
                imageview.startAnimation(animation);
            }
        }
}
```

(4) src\paint_2dtest 包下的 MainActivity.java 文件采用默认代码。

(5) 打开 res\layout 目录下的 main.xml 布局文件,在文件中调用自定义 PathMenuView,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tutor = "http://schemas.android.com/apk/res/fs.paint_2dtest"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#ffcc66" >
<fs.paint_2dtest.PathMenuView
        android:id = "@ + id/text"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        tutor:position = "right_bottom" />
</LinearLayout>
```

运行程序,效果如图 4-18(a)所示,当单击右下角的 Android 图标按钮时,效果如图 4-18(b)所示,再单击该图标,效果如图 4-18(a)所示。



(a) 自定义菜单排列效果



(b) 菜单收缩效果

图 4-18 超仿 Path 菜单效果 1

当把 main.xml 文件中的 tutor 属性设为 left_bottom,运行程序,效果如图 4-19 所示。



图 4-19　超仿 Path 菜单 2

4.14　视图的综合经典案例

在前面学习 Gallery 时,可以看到 Gallery 中保存了一系列图片,图片在容器中可以是横向排列或是垂直排列,而在一些手机应用中,经常可以看到一些立体的图片在类似 Gallery 的容器中显示,这就需要用到镜像特效来使图片变得更加立体。

镜像特效,顾名思义,在图片下方加上本身图片的"倒影",同时生成的"倒影"经过模糊和适当的压缩等处理,使得原来的图片像是放在一面镜子上,从而让图片本身看起来更加立体,这就是镜像特效。为了完成镜像特效,结合之前所学,需要重写 Gallery 类、自己的 ImageView 类和适配器 Adapter 类。

要实现这样的镜像特效,其具体操作步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 Mirror。

(2) 实现相簿显示。在相簿显示中,实现的图像显示不再是同一水平线上的显示,对于两侧的图片进行一定角度的旋转以及远小近大的处理。在 src/fs.mirror 包下新建一个 MirrorGallery.java 文件,在文件中实现做图像 3D 效果处理,其代码为:

```
package fs.mirror;
import android.content.Context;
import android.graphics.Camera;
import android.graphics.Matrix;
import android.util.AttributeSet;
import android.view.View;
import android.view.animation.Transformation;
import android.widget.Gallery;
import android.widget.ImageView;
public class MirrorGallery extends Gallery {
  //用来做类 3D 效果处理,如 z 轴方向上的平移、绕 y 轴的旋转等
```

```
private Camera mCamera = new Camera();
//图片绕 y 轴最大旋转角度,也就是屏幕最边上那两张图片的旋转角度
private int mMaxRotationAngle = 60;
//图片在 z 轴平移的距离,视觉上看起来就是放大缩小的效果
private int mMaxZoom = -380;
private int mCoveflowCenter;
private boolean mAlphaMode = true;
private boolean mCircleMode = false;
//构造函数
public MirrorGallery(Context context) {
    super(context);
    this.setStaticTransformationsEnabled(true);
}
public MirrorGallery(Context context, AttributeSet attrs) {
    super(context, attrs);
    this.setStaticTransformationsEnabled(true);
}
public MirrorGallery(Context context, AttributeSet attrs, int defStyle){
    super(context, attrs, defStyle);
    this.setStaticTransformationsEnabled(true);
}
public int getMaxRotationAngle() {
    return mMaxRotationAngle;
}
public void setMaxRotationAngle(int maxRotationAngle) {
    mMaxRotationAngle = maxRotationAngle;
}
public boolean getCircleMode() {
    return mCircleMode;
}
public void setCircleMode(boolean isCircle) {
    mCircleMode = isCircle;
}
public boolean getAlphaMode() {
    return mAlphaMode;
}
public void setAlphaMode(boolean isAlpha) {
    mAlphaMode = isAlpha;
}
public int getMaxZoom() {
    return mMaxZoom;
}
public void setMaxZoom(int maxZoom) {
    mMaxZoom = maxZoom;
}
private int getCenterOfCoverflow() {
    return (getWidth() - getPaddingLeft() - getPaddingRight()) / 2
            + getPaddingLeft();
}
//得到子对象的中线
private static int getCenterOfView(View view) {
    return view.getLeft() + view.getWidth() / 2;
}
protected boolean getChildStaticTransformation(View child, Transformation t) {
    final int childCenter = getCenterOfView(child);
```

```
            final int childWidth = child.getWidth();
            int rotationAngle = 0;
            t.clear();
            t.setTransformationType(Transformation.TYPE_MATRIX);
            if (childCenter == mCoveflowCenter) {
                transformImageBitmap((ImageView) child, t, 0);
            } else {
            rotationAngle = (int) (((float) (mCoveflowCenter - childCenter) / childWidth) *
    mMaxRotationAngle);
                if (Math.abs(rotationAngle) > mMaxRotationAngle) {
                    rotationAngle = (rotationAngle < 0) ? - mMaxRotationAngle
                            : mMaxRotationAngle;
                }
                transformImageBitmap((ImageView) child, t, rotationAngle);
            }
            return true;
        }
        protected void onSizeChanged(int w, int h, int oldw, int oldh) {
            mCoveflowCenter = getCenterOfCoverflow();
            super.onSizeChanged(w, h, oldw, oldh);
        }
        private void transformImageBitmap(ImageView child, Transformation t,
                int rotationAngle) {
            mCamera.save();
            final Matrix imageMatrix = t.getMatrix();
            final int imageHeight = child.getLayoutParams().height;
            final int imageWidth = child.getLayoutParams().width;
            final int rotation = Math.abs(rotationAngle);
            mCamera.translate(0.0f, 0.0f, 100.0f);
            //作为视图的角度变得更小,放大
            if (rotation <= mMaxRotationAngle) {
                float zoomAmount = (float) (mMaxZoom + (rotation * 1.5));
                mCamera.translate(0.0f, 0.0f, zoomAmount);
                if (mCircleMode) {
                    if (rotation < 40)
                        mCamera.translate(0.0f, 155, 0.0f);
                    else
                        mCamera.translate(0.0f, (255 - rotation * 2.5f), 0.0f);
                }
                if (mAlphaMode) {
                    ((ImageView) (child)).setAlpha((int) (255 - rotation * 2.5));
                }
            }
            mCamera.rotateY(rotationAngle);
            mCamera.getMatrix(imageMatrix);
            imageMatrix.preTranslate(-(imageWidth / 2), -(imageHeight / 2));
            imageMatrix.postTranslate((imageWidth / 2), (imageHeight / 2));
            mCamera.restore();
        }
    }
```

在以上代码中，Camera 是声明 android.graphics 包中的 Camera 类，是用作图像 3D 效果处理的，如 Z 轴方向上的平移、绕 Y 轴的旋转等。

变量 mMaxRoationAngle 用来记录一个旋转的角度，已知 Gallery 中一般很少只显示一个图片，那么除了正中央的图片外，还有两侧的图片，为了能让中央图片更加立体化，可以让两

侧图片适当向内侧倾斜一个角度，即以屏幕垂直线为 Y 轴的话，这个角度是绕 Y 轴旋转的角度。

变量 mMaxZoon 是让两侧图片的高度向内逐步缩小，视觉上给人一种远小近大的感觉。Android 中的 Z 轴是垂直屏幕向外的，所以这里的值为负值。

transformImageBitmap 函数中完成的工作主要是让两侧的图片向内旋转。左侧图片和右侧图片角度一样，只是正负不同。

（3）实现镜像效果。此时图像已实现了基本的正向显示，并且在正向显示中具有立体效果。接着，实现其镜面的效果。在 src/fs. mirror 包下新建一个 ImageG. java 文件，其代码为：

```
package fs.mirror;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Bitmap.Config;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.LinearGradient;
import android.graphics.Matrix;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.widget.ImageView;
import android.graphics.PorterDuffXfermode;
import android.graphics.PorterDuff.Mode;
import android.graphics.Shader.TileMode;
import android.graphics.drawable.BitmapDrawable;
public class ImageG extends ImageView {
    //是否为 Reflection 模式
    private boolean mReflectionMode = true;
    public ImageG(Context context) {
        super(context);
    }
    public ImageG(Context context, AttributeSet attrs) {
        super(context, attrs);
        //取得原始图片的 bitmap 并重画
        Bitmap originalImage = ((BitmapDrawable) this.getDrawable())
                .getBitmap();
        DoReflection(originalImage);
    }
    public ImageG(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        Bitmap originalImage = ((BitmapDrawable) this.getDrawable())
                .getBitmap();
        DoReflection(originalImage);
    }
    public void setReflectionMode(boolean isRef) {
        mReflectionMode = isRef;
    }
    public boolean getReflectionMode() {
        return mReflectionMode;
    }
    //只重写了 setImageResource，和构造函数做同样的事情
    @Override
    public void setImageResource(int resId) {
        Bitmap originalImage = BitmapFactory.decodeResource(getResources(),resId);
```

```
            DoReflection(originalImage);
        }
        private void DoReflection(Bitmap originalImage) {
            final int reflectionGap = 4; //原始图片和反射图片中间的间距
            int width = originalImage.getWidth();
            int height = originalImage.getHeight();
            //反转
            Matrix matrix = new Matrix();
            matrix.preScale(1, -1);
            /* reflectionImage 就是下面透明的那部分,可以设置它的高度为原始的 3/4,这样效果会更
好些 */
            Bitmap reflectionImage = Bitmap.createBitmap(originalImage, 0, 0,
                    width, (height/4) * 3, matrix, false);
            //创建一个新的 bitmap,高度为原来的两倍
            Bitmap bitmapWithReflection = Bitmap.createBitmap(width,
                    (height + height), Config.ARGB_8888);
            Canvas canvasRef = new Canvas(bitmapWithReflection);
            //先画原始的图片
            canvasRef.drawBitmap(originalImage, 0, 0, null);
            //画间距
            Paint deafaultPaint = new Paint();
            canvasRef.drawRect(0, height, width, height + reflectionGap,deafaultPaint);
            //画被反转以后的图片
            canvasRef.drawBitmap(reflectionImage, 0, height + reflectionGap, null);
            //创建一个渐变的蒙版放在下面被反转的图片上面
            Paint paint = new Paint();
            LinearGradient shader = new LinearGradient(0,
                    originalImage.getHeight(), 0, bitmapWithReflection.getHeight()
                            + reflectionGap, 0x80ffffff, 0x00ffffff, TileMode.CLAMP);
            //利用笔画使用这个(线性渐变)
            paint.setShader(shader);
            //设置传输模式
            paint.setXfermode(new PorterDuffXfermode(Mode.DST_IN));
            //线性渐变的矩形
            canvasRef.drawRect(0, height, width, bitmapWithReflection.getHeight()
                    + reflectionGap, paint);
            //调用 ImageView 中的 setImageBitmap
            this.setImageBitmap(bitmapWithReflection);
        }
    }
```

在 DoReflection 函数中,reflectionImage 创建一个新图片,即倒影。为了使创建的倒影更加逼真,将高度设置成原来图片高度的 3/4,这是为了模仿水和空气不同的折射率造成人视觉上对水中物体感觉比空气中原始物体更短。

bitmapWithReflection 用来画出原始图片和"倒影"。shader 是线性蒙化方法,为"倒影"加上阴影,使得"倒影"更加真实。

(4) 实现显示数据的适配器。在 src\fs.mirror 包下新建一个 AdapterG.java 文件,其代码为:

```
package fs.mirror;
import android.content.Context;
import android.graphics.drawable.BitmapDrawable;
import android.view.View;
import android.view.ViewGroup;
```

```
import android.widget.BaseAdapter;
import android.widget.ImageView;
public class AdapterG extends BaseAdapter {
    int mGalleryItemBackground;
    private Context mContext;
    private Integer[] Imgid = { R.drawable.a01, R.drawable.a02, R.drawable.a03,
            R.drawable.a04, R.drawable.a05 };
    public AdapterG (Context c) {
        mContext = c;
    }
    public int getCount() {
        return Imgid.length;
    }
    public Object getItem(int position) {
        return position;
    }
    public long getItemId(int position) {
        return position;
    }
    public View getView(int position, View convertView, ViewGroup parent) {
        ImageG i = new ImageG(mContext);

        i.setImageResource(Imgid[position]);
        i.setLayoutParams(new MirrorGallery.LayoutParams(160, 240));
        i.setScaleType(ImageView.ScaleType.CENTER_INSIDE);
        //确保设置抗锯齿,否则将得到锯齿
        BitmapDrawable drawable = (BitmapDrawable) i.getDrawable();
        drawable.setAntiAlias(true);
        return i;
    }
    public float getScale(boolean focused, int offset) {
        return Math.max(0, 1.0f / (float) Math.pow(2, Math.abs(offset)));
    }
}
```

在代码中,setLayoutParams 函数中使用了新的 Gallery 类。drawable.setAntiAlias(true)函数为设置抗锯齿。

(5) 启动 Activity 活动。打开 src\fs.mirror 包下的 MainActivity.java 文件,其代码为:

```
package fs.mirror;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
public class MainActivity extends Activity {
    public MirrorGallery gallery;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        gallery = (MirrorGallery)findViewById(R.id.gallery);
        gallery.setAdapter(new AdapterG(this));
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
```

```
        }
    }
```

（6）自定义控件布局。此处需要用到新的自定义 MirrorGallery 控件。对于自定义控件使用该控件定义类的全路径来标识，本例中使用 fs. mirror. MirrorGallery 来指定程序所在包中的新自定义控件。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:background="#aabbcc">
<fs.mirror.MirrorGallery
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/gallery"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />
</RelativeLayout>
```

第 5 章　Android 动画

动画(Animations)是一个实现 Android UI 界面动画效果的 API,提供了一系列的动画效果,可以进行旋转、缩放、淡入淡出等,这些效果应用在绝大多数的控件中。

动画片从总体上可以分为两大类:

- 补间动画(Tweened Animations):Animations 类提供了旋转、移动、伸展和淡出等效果。
- 帧动画(Frame-by-frame Animations):Animations 类可以创建一个 Drawable 序列,这些 Drawable 可以按照指定的时间间歇一个一个地显示。

5.1　补间动画

补间动画给出两个关键帧,通过一些算法将给定属性值在给定的时间内在两个关键帧间进行渐变。

补间动画只能应用于 View 对象,而且只支持一部分属性,如支持缩放旋转而不支持背景颜色的改变。对于补间动画,它只是改变了 View 对象绘制的位置,而没有改变 View 对象本身。例如,有一个 Button,坐标为(100,100),Width 为 200,Height 为 50,而有一个动画使其变为 Width 为 100,Height 为 100,就会发现动画过程中触发按钮点击的区域仍是(100,100)—(300,150)。

补间动画就是一系列 View 形状的变换,如大小的缩放、透明度的改变、位置的改变。动画的定义既可以用代码定义也可以用 XML 定义,当然,建议用 XML 定义。可以给一个 View 同时设置多个动画,如从透明至不透明的淡入效果、从小到大的放大效果。这些动画可以同时进行,也可以在一个完成之后开始另一个。

用 XML 定义的动画放在/res/anim/文件夹内,XML 文件的根元素可以为<alpha>、<scale>、<translate>、<rotate>,interpolator 元素或<set>(表示以上几个动画的集合,set 可以嵌套)。默认情况下,所有动画是同时进行的,可以通过 startOffset 属性设置各个动画的开始偏移(开始时间)来达到动画顺序播放的效果。可以通过设置 interpolator 属性改变动画渐变的方式,如 AccelerateInterpolator,开始时慢,然后逐渐加快。默认为 AccelerateDecelerateInterpolator。

Tween Animation 动画的 XML 使用格式为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android"
    android:interpolator = "@[package:]anim.interpolator_resource"
    android:shareInterpolator = ["true"|"false"]>
    < alpha android:fromAlpha = "float"
```

```
            android:toAlpha = "float"/>
      < scale android:fromXScale = "float"
            android:toXScale = "float"
            android:fromYScale = "float"
            android:toXScale = "float"
            android:pivotY = "float"/>
      < translate android:fromX = "float"
          android:toX = "float"
          android:fromY = "float"
          android:toY = "float"/>
      < rotate android:fromDegrees = "float"
            android:toDegrees = "float"
            android:pivotX = "float"
            android:pivotY = "float"/>
      < set >...
          </set >
</set >
```

从 XML 配置文件中可以看到，补间动画可以支持 Alpha 淡入淡出、Scale 缩放、Translate 移动和 Rotate 旋转等。

帧动画的常用方法主要如下：

- setDuration(long durationMills)：设置动画持续时间(单位为毫秒)。
- setFillAfter(Boolean fillAfter)：如果 fillAfter 的值为 true，则动画执行后，控件将停留在执行结束的状态。
- setFillBefore(Boolean fillBefore)：如果 fillBefore 的值为 true，则动画执行后，控件将回到动画执行之前的状态。
- setStartOffSet(long startOffSet)：设置动画执行之前的等待时间。
- setRepeatCount(int repeatCount)：设置动画重复执行的次数。

5.1.1 淡入淡出动画

在 Android 中提供了 Alpha 对象用于实现图像的淡出淡入效果，主要控制的是透明度的变化。Alpha 中定义可实现淡出淡入效果的关键属性如下：

- fromAlpha：表示动画起始时的透明度，0.0 表示完全透明，1.0 表示完全不透明，其取值在 0.0～1.0 之间 float 数据类型的数字。
- toAlpha：表示动画结束时的透明度，其取值在 0.0～1.0 之间的 float 数据类型的数字。
- duration：动画持续时间，单位为毫秒。

在 xml 中定义 alpha 动画的形式为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android">
    <!-- 透明度控制动画效果 alpha
            浮点型值:
                fromAlpha: 属性为动画起始时透明度
                toAlpha: 属性为动画结束时透明度
                        说明:
                    0.0 表示完全透明
                    1.0 表示完全不透明
              以上值取 0.0～1.0 之间的 float 数据类型的数字
```

```
            长整型值:
                duration: 属性为动画持续时间
                        说明:
                            时间以毫秒为单位  -->
    <alpha
        android:fromAlpha = "0.1"
        android:toAlpha = "1.0"
        android:duration = "5000"/>
</set>
```

下面通过一个案例来演示 Alpha 动画的用法。

【例 5-1】 利用 Alpha 控件实现动画切换(淡入淡出效果)。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li5_1Alpha。

(2) 创建第二个 Activity,实现 Activity 的切换效果。在 src\fs. li5_1alpha 包下创建 OtherActivity. java 文件,其代码为:

```
package fs.li5_1alpha;
import android.app.Activity;
import android.os.Bundle;
public class OtherActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //TODO 自动存根法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.other);
        }
}
```

(3) 打开 src\fs. li5_1alpha 包下的 MainActivity. java 文件,实现 openActivity 方法,用来打开新的 Activity,其代码为:

```
package fs.li5_1alpha;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
public class MainActivity extends Activity {
/** 第一次调用活动. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
/**
 * 功能:打开新的 Activity
 */
public void openActivity(View v){
    Intent intent = new Intent(this, OtherActivity.class);
    startActivity(intent);
    /* 屏幕动画淡入淡出效果切换,调用 anim 文件夹中创建的 enteralpha(进入动画)和 exitalpha
(淡出动画)两个动画(注意,两个 xml 文件命名不能有大写字母) */
    //如果想定义其他动画效果,只需要改变 enteralpha 和 exitalpha 两个文件
    this.overridePendingTransition(R.anim.enteralpha,R.anim.exitalpha);
 }
}
```

(4) 打开 res\layout 目录下的 main. xml 文件,添加“打开新 Activity”按钮,点击事件调用 MainActivity 类中 openActivity 方法,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#aabbcc" >
< Button
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "@string/hello_world"
    android:onClick = "openActivity"/>
</LinearLayout >
```

(5) 在 res\layout 目录创建 other. xml 文件,在文件中定义 OtherActivity,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#6600FF" >
< TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "打开新的 Activity" />
</LinearLayout >
```

(6) 在 res 文件夹下创建 anim 文件夹,其中再创建淡入淡出动画效果文件,分别为 enteralpha. xml 和 exitalpha. xml。注意,文件名只能为小写 a～z,数字 0～9 和下划线。

enteralpha. xml 文件的代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android"
    android:shareInterpolator = "false">
    <!-- 动画效果: 从看不见到可以看见,也就是 0 到 1,变幻时间为 5000 毫秒 -- >
    < alpha
        android:fromAlpha = "0"
        android:toAlpha = "1"
        android:duration = "5000"/>
</set >
```

exitalpha. xml 文件的代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android"
    android:shareInterpolator = "false">
    <!-- 退出动画效果: 从可以看见到不可以看见,也就是 1.0 到 0,变幻时间为 5000 毫秒 -- >
< alpha
    android:fromAlpha = "1.0"
    android:toAlpha = "0"
    android:duration = "5000"/>
</set >
```

(7) 打开 AndroidManifest. xml 文件,在文件中添加设置权限。修改后代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
```

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="fs.li5_1alpha"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="fs.li5_1alpha.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".OtherActivity"
            android:label="新窗口"></activity>
    </application>
</manifest>
```

5.1.2 缩放动画

在 Android 中提供了 Scale 控件实现动画的缩放。Scale 中定义可实现缩放的关键属性有：

- fromXScale：起始时 x 坐标的尺寸，设置为 1.0 说明是整个图片 x 轴的长度。
- toXScale：结束时 x 坐标的尺寸，设置为 0.0 说明整个图片 x 轴完全收缩到无。
- fromYScale：起始时 y 坐标的尺寸，设置为 1.0 说明是整个图片 y 轴的长度。
- toYScale：结束时 y 坐标的尺寸，设置为 1.0 说明是在收缩时 y 轴的长度保持不变。
- pivotX：动画在 X 轴方向缩放的中心点，值取 0%～100%或 0%p～100%p，50%为相对于自己的中心位置，50%p 表示相对于父控件的中心位置。
- pivotY：动画在 Y 轴方向缩放的中心点，值取 0%～100%或 0%p～100%p，50%为相对于自己的中心位置，50%p 表示相对于父控件的中心位置。
- duration：是设置的动画执行时间。
- interpolator：指定一个动画的插入器。Interpolator 定义一个动画的变化率，这使得基本的动画效果(Alpha、Scale、Translate、Rotate)得以加速、减速、重复等。Android 提供了几个 Interpolator 子类，实现了不同的速度曲线。

◇ linear_interpolator：使动画以均匀的速率改变。

◇ cycle_interpolator：动画循环播放特定的次数，速率改变沿着正弦曲线。

◇ accelerate_decelerate_interpolator：在动画开始与结束的地方速率改变比较慢，在中间时加速。

◇ accelerate_interpolator：在动画开始的地方速率改变比较慢，然后开始加速。

◇ decelerate_interpolator：在动画开始的地方改变比较慢，然后开始减速。

- zAdjustment：定义动画的 Z 轴方向的位置，值 normal 保持位置不变，top 保持在最上层，bottom 保持在最下层。
- repeatCount：动画的重复次数。
- repeatMode：定义重复的模式，其中 restart 表示重新开始，reverse 先倒退再执行，倒退也算一次。
- startOffset：动画之间的时间间隔，即从上次动画停多少时间开始执行下一个动画。

在 XML 中定义 alpha 动画的形式为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android"
    android:interpolator = "@android:anim/accelerate_interpolator">
   <!--
       起始 x 轴坐标
           止 x 轴坐标
           始 y 轴坐标
           止 y 轴坐标
           轴的坐标
           轴的坐标  -->
   < scale
       android:fromXScale = "1.0"
       android:toXScale = "0.0"
       android:fromYScale = "1.0"
       android:toYScale = "0.0"
       android:pivotX = "50 % "
       android:pivotY = "50 % "
       android:duration = "1000"/>
</set >
```

下面通过一个案例来演示动画的缩放。

【例 5-2】 利用 scale 控件实现动画的缩放。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li5_2Scale。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明一个 ImageView 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = " # aabbcc">
< ImageView
    android:id = "@ + id/imgView"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_marginTop = "200px"
    android:src = "@drawable/a1"/>
</LinearLayout >
```

(3) 打开 src\fs.li5_2scale 包下的 MainActivity.java 文件，在文件中实现图像的翻转及缩放，其代码为：

```
package fs.li5_2scale;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
```

```
import android.view.View.OnClickListener;
import android.view.animation.AlphaAnimation;
import android.view.animation.Animation;
import android.view.animation.AnimationSet;
import android.view.animation.AnimationUtils;
import android.view.animation.ScaleAnimation;
import android.widget.ImageView;
public class MainActivity extends Activity {
    /** 第一次调用活动. */
    private ImageView imgView;
    //声明一个 boolean 用来切换背面和正面
    private boolean bool = false;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        imgView = (ImageView) findViewById(R.id.imgView);
        //给 ImageView 添加点击事件
        imgView.setOnClickListener(new ImgViewListener());
    }
    class ImgViewListener implements OnClickListener {
        @Override
        public void onClick(View v) {
            //TODO 自动存根法
            //通过 AnimationUtils 得到动画配置文件(/res/anim/back.xml)
  Animation animation = AnimationUtils.loadAnimation(MainActivity.this, R.anim.back);
           animation.setAnimationListener(new Animation.AnimationListener() {
                @Override
                public void onAnimationStart(Animation animation) {
                }
                @Override
                public void onAnimationRepeat(Animation animation) {
                }
                @Override
                public void onAnimationEnd(Animation animation) {
                    if(bool){
                        imgView.setImageResource(R.drawable.a1);
                        bool = false;
                    }else {
                        imgView.setImageResource(R.drawable.a03);
                        bool = true;
                    }
                  /* 通过 AnimationUtils 得到动画配置文件(/res/anim/front.xml),然后再把动画
交给 ImageView */
                    imgView.startAnimation(AnimationUtils.loadAnimation(MainActivity.this,
R.anim.front));
                }
            });
            imgView.startAnimation(animation);
        }
    }
}
```

(4) 在 res 目录下创建一个 anim 文件,在文件中创建两个文件,分别为 back.xml 及 front.xml 文件。

back.xml 文件实现 Scale 的定义,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
```

```
<set xmlns:android = "http://schemas.android.com/apk/res/android"
      android:interpolator = "@android:anim/accelerate_interpolator">
      <scale
         android:fromXScale = "1.0"
         android:toXScale = "0.0"
         android:fromYScale = "1.0"
         android:toYScale = "1.0"
         android:pivotX = "50%"
         android:pivotY = "50%"
         android:duration = "150"/>
</set>
```

front.xml 文件，以 back.xml 文件属性相同，只不过是以圆心点为轴向两边伸展到完全展开，其代码为：

```
<?xml version = "1.0" encoding = "utf-8"?>
<set xmlns:android = "http://schemas.android.com/apk/res/android"
      android:interpolator = "@android:anim/accelerate_interpolator">
      <scale
         android:fromXScale = "0.0"
         android:toXScale = "1.0"
         android:fromYScale = "1.0"
         android:toYScale = "1.0"
         android:pivotX = "50%"
         android:pivotY = "50%"
         android:duration = "150"/>
</set>
```

运行程序，效果如图 5-1 所示。当单击屏幕上的图片时，图片进行切换、翻转及缩放。



图 5-1　图像的翻转

5.1.3　旋转动画

在 Android 中提供了 Rotate 对象用于实现图像的旋转。旋转和缩放都需要指定中心点，同样在取值时要指定是相对于父控件还是相对于自己，50% 表示的是自己的中心。Rotate 中定义可实现旋转的关键属性有：

- fromDegrees：动画开始时的角度。
- toDegrees：动画结束时旋转的角度，可以大于 360 度。

在 XML 中定义 rotate 动画的形式为：

```
<?xml version = "1.0" encoding = "utf-8"?>
<set xmlns:android = "http://schemas.android.com/apk/res/android"
     android:interpolator = "@android:anim/accelerate_interpolator">
     <!--
         fromDegrees: 开始的角度
         toDegrees: 结束的角度, + 表示是正的
         pivotX: 用于设置旋转时的 x 轴坐标
         例
            (1)当值为"50",表示使用绝对位置定位
            (2)当值为"50%",表示使用相对于控件本身定位
            (3)当值为"50%p",表示使用相对于控件的父控件定位
         pivotY: 用于设置旋转时的 y 轴坐标      -->
```

```
    <rotate
        android:fromDegrees = "0"
        android:toDegrees = " + 360"
        android:pivotX = "50 % "
        android:pivotY = "50 % "
        android:duration = "1000"/>
</set>
```

下面通过案例使用 Rotate 演示动画的旋转。

【例 5-3】 利用 Rotate 控件实现动画的旋转。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li5_3Rotate。

(2) 打开 res\layout 目录下的 main. xml 布局文件，在文件中声明两个 Button 控件及一个 ImageView 控件，其代码为：

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = " # aabbcc">
    <Button
        android:id = "@ + id/bt1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignRight = "@ + id/button1"
        android:layout_below = "@ + id/button1"
        android:layout_marginTop = "26dp"
        android:text = "开始动画" />
    <Button
        android:id = "@ + id/bt2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_below = "@ + id/bt1"
        android:text = "取消动画" />
<ImageView
        android:id = "@ + id/imgView"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/bt1"
        android:layout_below = "@ + id/bt2"
        android:layout_marginTop = "67dp"
        android:src = "@drawable/c1" />
</LinearLayout>
```

(3) 打开 src\fs. li5_3rotate 包下的 MainActivity. java 文件，在文件中实现动画的旋转及动画的取消，其代码为：

```
package fs.li5_3rotate;
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.view.View;
import android.view.View.OnClickListener;
```

```
import android.view.animation.Animation;
import android.view.animation.RotateAnimation;
import android.widget.Button;
import android.widget.ImageView;
public class MainActivity extends Activity {
    ImageView image;
    Button start;
    Button cancel;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        image = (ImageView) findViewById(R.id.imgView);
        start = (Button) findViewById(R.id.bt1);
        cancel = (Button) findViewById(R.id.bt2);
        /* 设置旋转动画 */
        final RotateAnimation animation = new RotateAnimation(0f,360f,Animation.RELATIVE_TO_
SELF, 0.5f,Animation.RELATIVE_TO_SELF,0.5f);
        animation.setDuration(3000);                        //设置动画持续时间
        /* 常用方法 */
        start.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                image.setAnimation(animation);
                /* 开始动画 */
                animation.startNow();
                }
            });
        cancel.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                /* 结束动画 */
                animation.cancel();
            }
      });
    }
}
```

运行程序，效果如图 5-2 所示。当单击屏幕中的“开始动画”按钮时，即图像开始旋转，当单击屏幕中的“取消动画”按钮时，即图像旋转停止。



图 5-2　旋转图像

5.1.4　平移图像

在 Android 中提供了 Translate 对象用于实现图像的平移。Translate 中定义可实现平移的关键属性有：

- float fromXDelta 动画开始的点离当前 View x 坐标上的差值。
- float toXDelta 动画结束的点离当前 View x 坐标上的差值。
- float fromYDelta 动画开始的点离当前 View y 坐标上的差值。

- float toYDelta 动画开始的点离当前 View y 坐标上的差值。

在 XML 中定义 rotate 动画的形式为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android"
    android:interpolator = "@android:anim/accelerate_interpolator">
    <!--      始 x 轴坐标
              止 x 轴坐标
              始 y 轴坐标
              止 y 轴坐标      -->
    < translate
        android:fromXDelta = "0 % "
        android:toXDelta = "100 % "
        android:fromYDelta = "0 % "
        android:toYDelta = "100 % "
        android:duration = "2000"/>
</set >
```

下面通过一个案例演示 Translate 控件的用法。

【例 5-4】 使用 Translate 控件实现图像的平移。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li5_4Translate。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明两个 Button 控件和一个 ImageView 控件，其代码为：

```
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
    < Button
        android:id = "@ + id/bt1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignRight = "@ + id/button1"
        android:layout_below = "@ + id/button1"
        android:layout_marginTop = "26dp"
        android:text = "开始动画" />
    < Button
        android:id = "@ + id/bt2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_below = "@ + id/bt1"
        android:text = "取消动画" />
< ImageView
        android:id = "@ + id/imgView"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/bt1"
        android:layout_below = "@ + id/bt2"
        android:layout_marginTop = "67dp"
        android:src = "@drawable/b7" />
</LinearLayout >
```

(3) 打开 src\fs.li5_4translate 包下的 MainActivity.java 文件，在文件中实现图像的平移及取消图像，其代码为：

```
package fs.li5_4translate;
```

```
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.animation.Animation;
import android.view.animation.TranslateAnimation;
import android.widget.Button;
import android.widget.ImageView;
public class MainActivity extends Activity {
    ImageView image;
    Button start;
    Button cancel;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        image = (ImageView) findViewById(R.id.imgView);
        start = (Button) findViewById(R.id.bt1);
        cancel = (Button) findViewById(R.id.bt2);
        /* 设置位移动画,向右位移 150 */
        final TranslateAnimation animation = new TranslateAnimation(0, 150,0, 0);
        animation.setDuration(2000);                        //设置动画持续时间
        animation.setRepeatCount(2);                        //设置重复次数
        animation.setRepeatMode(Animation.REVERSE);         //设置反方向执行
        start.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                image.setAnimation(animation);
                /* 开始动画 */
                animation.startNow();
            }
        });
        cancel.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                /* 结束动画 */
                animation.cancel();
            }
        });
    }
}
```

运行程序,效果如图 5-3 所示。当单击屏幕中的“开始动画”按钮时,即图像开始平移,当单击屏幕中的“取消动画”按钮时,即图像平移停止。



图 5-3 图像的平移

5.1.5 补间动画经典案例

下面通过一个综合案例来演示补间动画。

【例 5-5】 实现图像的淡出淡入、缩放、旋转、平移等动画效果。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li5_5 Animation。

(2) 打开 res\layout 目录下的 main.xml 文件,在文件中声明一个 Button 控件和一个 ImageView 控件,其代码为:

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc" >
<ImageView
    android:src = "@drawable/g4"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:id = "@ + id/image"/>
<Button
    android:layout_below = "@id/image"
    android:text = "开始"
    android:id = "@ + id/button"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"/>
</RelativeLayout>
```

(3) 在 res 目录下创建一个 anim 文件,在文件中创建一个名为 item.xml 的文件,在文件中声明补间动画,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<set
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:interpolator = "@android:anim/linear_interpolator">
<!-- 透明度变化 -->
     <alpha
         android:fromAlpha = "1"
         android:toAlpha = "0"
         android:duration = "2000"/>
<!-- 缩放与扩大 -->
     <scale android:fromXScale = "1.0"
        android:toXScale = "0"
        android:fromYScale = "1.0"
        android:toYScale = "0"
        android:pivotX = "50%"
        android:pivotY = "50%"
        android:fillAfter = "true"
        android:duration = "2000"/>
<!-- 水平与垂直位移 -->
    <translate
        android:fromXDelta = "0"
        android:toXDelta = "130"
        android:fromYDelta = "0"
```

```
            android:toYDelta = " - 80"
            android:duration = "2000"/>
    <!-- 旋转 -->
        <rotate
            android:fromDegrees = "0"
            android:toDegrees = "360"
            android:pivotX = "50 % "
            android:pivotY = "50 % "
            android:duration = "2000"/>
    </set>
```

(4) 打开 src\fs. li5__5animation 包下的 MainActivity. java 文件，在文件中实现图像的补间动画，其代码为：

```
package fs. li5_5animation;
import android. app. Activity;
import android. os. Bundle;
import android. view. Menu;
import android. view. View;
import android. view. View. OnClickListener;
import android. view. animation. Animation;
import android. view. animation. AnimationUtils;
import android. widget. Button;
import android. widget. ImageView;
public class MainActivity extends Activity {
    private ImageView image;
    private Button button;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super. onCreate(savedInstanceState);
        setContentView(R. layout. main);
        final Animation anim = AnimationUtils. loadAnimation(this, R. anim. item);
        button = (Button)findViewById(R. id. button);
        image = (ImageView)findViewById(R. id. image);
        button. setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                //TODO 自动存根法
                image. startAnimation(anim);
            }
        });
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        getMenuInflater(). inflate(R. menu. main, menu);
        return true;
    }
}
```

运行程序，效果如图 5-4 所示，单击屏幕上的“开始”按钮，即图像同时实现图像的补间动画，效果如图 5-4(b)所示。



(a) 默认界面



(b) 图像的动画效果

图 5-4　补间动画

5.2　帧　动　画

帧动画是比较传统的动画方式，帧动画将一系列的图片文件像放电影般依次进行播放，帧动画主要用到的类是 AnimationDrawable，每个帧动画都是一个 AnimationDrawable 对象。

定义帧动画可以在代码中直接进行，也可以通过 XML 文件定义，定义帧动画的 XML 文件将存放在项目的 res/anim 目录下。在 XML 文件中指定了图片帧出现的顺序及每个帧的持续时间。在帧动画的 XML 文件中主要用到的标记及其属性如表 5-1 所示。

表 5-1　帧动画中标记及其属性说明

标记名称	属　性　值	说　明
<animation-list>	android:oneshot：如果设置为 true，则该动画只播放一次，然后停止在最后一帧	Frame Animation 的根标记，包含若干<item>标记
<item>	android:drawable：图片帧的引用； android:duration：图片帧的停留时间； android:visible：图片帧是否可见	每个<item>标记定义一个图片帧，其中包含图片资源的引用等属性

定义逐帧动画的 XML 语法格式为：

```
[html] view plaincopyprint?<?xml version = "1.0" encoding = "utf - 8"?>
< animation - list xmlns:android = "http://schemas.android.com/apk/res/android"
    android:oneshot = ["true" | "false"] >
    < item
        android:drawable = "@[package:]drawable/drawable_resource_name"
        android:duration = "integer" />
</animation - list >
```

下面通过一个案例来实现图像的帧动画。

【例 5-6】 用 Animation-list 实现图像的帧动画。其具体实现步骤为：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 li5_6FrameAnimation。

（2）打开 res\layout 目录下的 main. xml 文件，在文件中声明一个 ImageView 控件及三个 Button 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#aabbcc">
    < ImageView
            android:id = "@ + id/animationIV"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:padding = "5px"
            android:src = "@drawable/animation1"/>
    < Button
        android:id = "@ + id/buttonA"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:padding = "5px"
        android:text = "顺序显示" />
    < Button
        android:id = "@ + id/buttonB"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:padding = "5px"
        android:text = "停止" />
    < Button
        android:id = "@ + id/buttonC"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:padding = "5px"
        android:text = "倒序显示" />
</LinearLayout >
```

（3）在 res\drawable-mdpi 目录下创建两个文件，分别为 animation1. xml 及 animation2. xml。animation1. xml 文件用于顺序显示动画文件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<!-- 根标签为 animation - list,其中,oneshot 表示只展示一遍; 设置为 false 会不停的循环播放动
画; 通过 item 标签对动画中的每一张图片进行声明; android:duration 表示展示该图片所用的时间长
度 -- >
< animation - list
  xmlns:android = "http://schemas.android.com/apk/res/android"
  android:oneshot = "true" >
    < item
        android:drawable = "@drawable/ab"
        android:duration = "150"></item >
    < item
        android:drawable = "@drawable/ac"
        android:duration = "150"></item >
    < item
        android:drawable = "@drawable/ad"
        android:duration = "150"></item >
    < item
```

```
            android:drawable = "@drawable/ae"
            android:duration = "150"></item>
        < item
            android:drawable = "@drawable/af"
            android:duration = "150"></item>
        < item
            android:drawable = "@drawable/ag"
            android:duration = "150"></item>
</animation - list>
```

animation2. xml 文件用于倒序显示动画文件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<!- - 根标签为 animation - list,其中,oneshot 表示只展示一遍; 设置为 false 会不停地循环播放动画; 通过 item 标签对动画中的每一张图片进行声明; android:duration 表示展示该图片所用的时间长度 - ->
< animation - list
    xmlns:android = "http://schemas. android. com/apk/res/android"
    android:oneshot = "true">
        < item
            android:drawable = "@drawable/ag"
            android:duration = "150"></item>
        < item
            android:drawable = "@drawable/af"
            android:duration = "150"></item>
        < item
            android:drawable = "@drawable/ae"
            android:duration = "150"></item>
        < item
            android:drawable = "@drawable/ad"
            android:duration = "150"></item>
        < item
            android:drawable = "@drawable/ac"
            android:duration = "150"></item>
        < item
            android:drawable = "@drawable/ab"
            android:duration = "150"></item>
</animation - list>
```

(4) 打开 src\fs. li5_6frameanimation 包下的 MainActivity. java,在文件中实现帧动画,其代码为:

```
package fs. li5_6frameanimation;
import android. app. Activity;
import android. graphics. drawable. AnimationDrawable;
import android. os. Bundle;
import android. view. View;
import android. view. View. OnClickListener;
import android. view. Window;
import android. widget. Button;
import android. widget. ImageView;
public class MainActivity extends Activity
{
    private ImageView animationIV;
    private Button buttonA, buttonB, buttonC;
    private AnimationDrawable animationDrawable;
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
            super.onCreate(savedInstanceState);
            requestWindowFeature(Window.FEATURE_NO_TITLE);
            setContentView(R.layout.main);
            animationIV = (ImageView) findViewById(R.id.animationIV);
            buttonA = (Button) findViewById(R.id.buttonA);
            buttonB = (Button) findViewById(R.id.buttonB);
            buttonC = (Button) findViewById(R.id.buttonC);
            buttonA.setOnClickListener(new OnClickListener()
            {
                @Override
                public void onClick(View v) {
                    //TODO 自动存根法
                    animationIV.setImageResource(R.drawable.animation1);
                   animationDrawable = (AnimationDrawable) animationIV.getDrawable();
                    animationDrawable.start();
                }
            });
            buttonB.setOnClickListener(new OnClickListener()
            {
                @Override
                public void onClick(View v) {
                    //TODO 自动存根法
                   animationDrawable = (AnimationDrawable) animationIV.getDrawable();
                    animationDrawable.stop();
                }
            });
            buttonC.setOnClickListener(new OnClickListener()
            {
                @Override
                public void onClick(View v) {
                    //TODO 自动存根法
                    animationIV.setImageResource(R.drawable.animation2);
                   animationDrawable = (AnimationDrawable) animationIV.getDrawable();
                    animationDrawable.start();
                }
            });
        }
    }
```

运行程序,效果如图 5-5 所示。



图 5-5　图像的帧动画

很多实际的动画往往同时运行两个动画,在此要实现一个小游戏,需要让用户控制游戏中的主角移动——当主角移动时,不仅要控制它的位置改变,还应该在它移动时播放 Frame 动画来让用户感觉更“逼真”。

【例 5-7】　本实例利用 Tween 动画与 Frame 动画开发的“老鹰飞翔”的效果,在这个实例中,老鹰飞翔时的振翅效果为 Frame 动画,而老鹰飞翔时的位置为 Tween 动画。其实现操作步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 EagleFly。

(2) 在 res 根目录新建一个 anim 子目录文件夹，新建一个 laoyangfly. xml 文件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<!-- 定义动画循环播放 -->
< animation - list xmlns:android = "http://schemas.android.com/apk/res/android"
    android:oneshot = "false">
    < item
        android:drawable = "@drawable/ly1" android:duration = "120" />
    < item
        android:drawable = "@drawable/ly2" android:duration = "120" />
    < item android:drawable = "@drawable/ly3" android:duration = "120" />
    < item
        android:drawable = "@drawable/ly4" android:duration = "120" />
    < item
        android:drawable = "@drawable/ly5" android:duration = "120" />
    < item
        android:drawable = "@drawable/ly6" android:duration = "120" />
</animation - list >
```

(3) 打开 res\Layout 目录下的 main. java 文件，在文件中声明一个 ImageView 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout
xmlns:apk = "http://schemas.android.com/apk/res/android"
apk:orientation = "vertical"
apk:layout_width = "fill_parent"
apk:layout_height = "fill_parent">
< ImageView
apk:id = "@ + id/laoyangfly"
apk:layout_width = "wrap_content"
apk:layout_height = "wrap_content"
apk:background = "@anim/laoyangfly" />
</LinearLayout >
```

(4) 打开 src/fs. eaglefly 包下的 MainActivity. java 文件，在文件中实现补间动画与帧动画，其代码为：

```
package fs.eaglefly;
import java.util.Timer;
import java.util.TimerTask;
import android.app.Activity;
import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.animation.TranslateAnimation;
import android.widget.ImageView;
public class MainActivity extends Activity
{
    //记录老鹰 ImageView 当前的位置
    private float curX = 0;
    private float curY = 30;
```

```
    //记录老鹰 ImageView 下一个位置的坐标
    private AnimationDrawable animDance;
    float nextX = 0;
    float nextY = 0;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获取显示老鹰的 ImageView 组件
        final ImageView imageView = (ImageView)findViewById(R.id.laoyangfly);
        final Handler handler = new Handler()
        {
            @Override
            public void handleMessage(Message msg)
            {
                if (msg.what == 0x123)
                {
                    //横向上一直向右飞
                    if(nextX > 320)
                    {
                        curX = nextX = 0;
                    }
                    else
                    {
                        nextX += 8;
                    }
                    //纵向上可以随机上下
                    nextY = curY + (float)(Math.random() * 10 - 5);
                    //设置显示老鹰的 ImageView 发生位移改变
                    TranslateAnimation anim
                         = new TranslateAnimation(curX, nextX, curY, nextY);
                    curX = nextX;
                    curY = nextY;
                    anim.setDuration(200);
                    //开始位移动画
                    imageView.startAnimation(anim);
                }
            }
        };
        final AnimationDrawable butterfly = (AnimationDrawable)imageView
            .getBackground();

    imageView.setOnClickListener(new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                //开始播放老鹰振翅的逐帧动画
                butterfly.start();
                //通过定制器控制每 0.2 秒运行一次 TranslateAnimation 动画
                new Timer().schedule(new TimerTask()
                {
                    @Override
                    public void run()
```

```
                    {
                        handler.sendEmptyMessage(0x123);
                    }
                }, 0, 50);
            }
        });
    }
}
```

运行程序,单击图片,效果如图 5-6 所示。

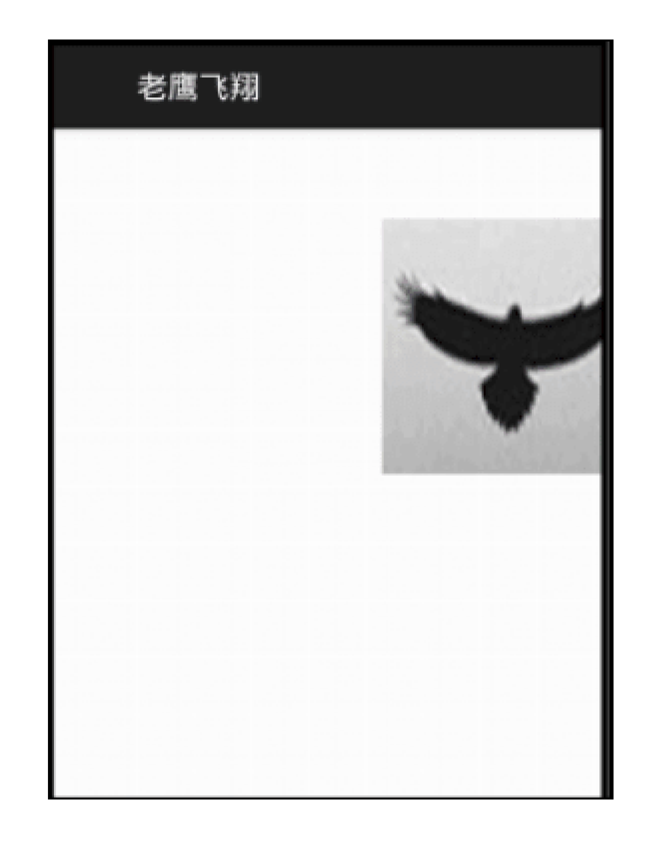


图 5-6 补间动画与帧动画实例

5.3 动画组件

本节介绍在 Android 中常用的动画组件(ViewAnimator),通过这些组件能方便地实现一组 View 动画。

动画组件的基类为 FrameLayout,作用是为 FrameLayout 里面的 View 切换提供动画效果。

一般不直接使用 ViewAnimator,而是使用它的子类 ViewFlipper 和 ViewSwitcher,其中 ViewSwitcher 的子类又包含了 ImageSwitcher 和 TextSwitcher。ViewFlipper 和 ViewSwitcher 的主要区别为 ViewSwitcher 最多能有两个子 View,而 ViewFlipper 可以有多个。

5.3.1 ViewSwitcher 组件

ImageSwitcher 是一个控制图片切换显示的组件,可添加图片切换动画,效果很好,适合做相册或动态展示图片。在 Android 中,还有一个类似的组件就是 TextSwitcher,它们的用法基本相同。

使用 ImageSwitcher 或 TextSwitcher 必须设置一个 ViewFactory,用来在 ViewSwitcher 中创建 View,因此需要实现 ViewSwitcher. ViewFactory 接口,最后通过 makeView()方法创建相应的 View,即 ImageSwitcher 对应 ImageView,TextSwitcher 对应 TextView。

下面通过一个案例来演示 ViewSwitcher 组件的用法。

【例 5-8】 通过 ViewSwitcher 来实现 Android 的分屏和左右滚动效果。

为了实现该效果,程序主界面考虑使用 ViewSwitcher 来组合多个 GridView,每个

GridView 代表一个屏幕的应用程序，GridView 中每个单元格显示一个应用程序图标的程序名。

其具体实现步骤为：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 li5_7ViewSwitcher。

（2）打开 res\layout 目录下的 main.xml 布局文件，在布局文件中定义了一个 ViewSwitcher 组件和两个按钮，这两个按钮分别用于控制该 ViewSwitcher 显示上一屏和下一屏的程序列表，其代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc">
  <!-- 定义一个 ViewSwitcher 组件 -->
  <ViewSwitcher
      android:id = "@ + id/viewSwitcher"
      android:layout_width = "fill_parent"
      android:layout_height = "fill_parent" />
  <!-- 定义滚动到上一屏的按钮 -->
<Button
     android:id = "@ + id/button_prev"
     android:layout_width = "wrap_content"
     android:layout_height = "wrap_content"
     android:layout_alignParentBottom = "true"
     android:layout_alignParentLeft = "true"
     android:onClick = "prev"
     android:text = "&lt;" />
<!-- 定义滚动下一屏的按钮 -->
<Button
     android:id = "@ + id/button_next"
     android:layout_width = "wrap_content"
     android:layout_height = "wrap_content"
     android:layout_alignParentBottom = "true"
     android:layout_alignParentRight = "true"
     android:onClick = "next"
     android:text = "&gt;"/>
</RelativeLayout>
```

（3）打开 src\fs.li5_7viewswitcher 包下的 MainActivity.java 文件，该文件的重点在于为该 ViewSwitcher 设置 ViewFactory 对象，并且当用户单击“＜”和“＞”两个按钮时控制 ViewSwitcher 显示“上一屏”和“下一屏”的应用程序。本实例的关键就是根据用户单击按钮来动态计算该 BaseAdapter 应该显示哪些程序列表，其代码为：

```
package fs.li5_7viewswitcher;
import java.util.ArrayList;
import android.os.Bundle;
import android.app.Activity;
import android.graphics.drawable.Drawable;
```

```
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.View;
import android.view.ViewGroup;
import android.widget.*;
import android.widget.ViewSwitcher.ViewFactory;
public class MainActivity extends Activity {
    //定义一个常量,用于显示每屏显示的应用程序数
    public static final int NUMBER_PER_SCREEN = 12;
    //代表应用程序的内部类
    public static class DataItem
    {
        //应用程序名称
        public String dataName;
        //应用程序图标
        public Drawable drawable;
    }
    //保存系统所有应用程序的 List 集合
    private ArrayList<DataItem> items = new ArrayList<DataItem>();
    //记录当前正在显示第几屏的程序
    private int screenNo = -1;
    //保存程序所占的总屏数
    private int screenCount;
    ViewSwitcher switcher;
    //创建 LayoutInflater 对象
    LayoutInflater inflater;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        inflater = LayoutInflater.from(MainActivity.this);
        //创建一个包含 40 个元素的 List 集合,用于模拟 40 应用程序
        for(int i = 0;i < 40;i++)
        {
        String label = "" + i;
        Drawable drawable = getResources().getDrawable(R.drawable.ic_launcher);
        DataItem item = new DataItem();
        item.dataName = label;
        item.drawable = drawable;
        items.add(item);
        }
     //计算应用程序所占的总屏数
     //如果应用程序的数量能整除 NUMBER_PER_SCREEN,除法的结果就是总屏数
     //如果不能整除,总屏数应该是除法的结果再加 1
       screenCount = items.size() % NUMBER_PER_SCREEN == 0?items.size()/NUMBER_PER_SCREEN:
            items.size()/NUMBER_PER_SCREEN + 1;
        switcher = (ViewSwitcher)findViewById(R.id.viewSwitcher);
        switcher.setFactory(new ViewFactory(){
             //实际上就是返回一个 GridView 组件
            @Override
            public View makeView() {
                //TODO 自动存根法
                //加载 R.layout.slidelistview 组件,实际上就是一个 GridView 组件
                return inflater.inflate(R.layout.slidelistview, null);
            }
```

```
        });
        next(null);
    }
    public void next(View v)
    {
        if(screenNo < screenCount - 1)
        {
            screenNo++;
            //为 ViewSwitcher 的组件显示过程设置动画
            switcher.setInAnimation(this,R.anim.slide_in_right);
            //为 ViewSwitcher 的组件隐藏过程设置动画
            switcher.setOutAnimation(this,R.anim.slide_out_left);
            //控制下一屏将要显示的 GridView 对应的 Adapter
            ((GridView) switcher.getNextView()).setAdapter(adapter);
            //单击右边显示下一屏
            //学习手势检测后,也可通过手势检测实现显示下一屏
            switcher.showNext();
        }
    }
    public void prev(View v)
    {
        if(screenNo > 0)
        {
            screenNo--;
            //为 ViewSwitcher 的组件显示过程设置动画
            switcher.setInAnimation(this,R.anim.slide_in_left);
            //为 ViewSwitcher 的组件隐藏过程设置动画
            switcher.setOutAnimation(this,R.anim.slide_out_right);
            //控制下一屏将要显示的 GridView 对应的 Adapter
            ((GridView)switcher.getNextView()).setAdapter(adapter);
            //单击左边按钮,显示上一屏,当然可以采用手势
            //学习手势检测后,也可通过手势检测实现显示上一屏
            switcher.showPrevious();
        }
    }
    //该 BaseAdapter 负责为每屏显示的 GridView 提供列表项
    private BaseAdapter adapter = new BaseAdapter()
    {
        @Override
        public int getCount() {
     //TODO 自动存根法
     //如果已经到了最后一屏,且应用程序的数量不能整除 NUMBER_PER_SCREEN if(screenNo ==
screenCount - 1&&items.size() % NUMBER_PER_SCREEN!= 0)
            {
     //最后一屏显示的程序数为应用程序的数量对 NUMBER_PER_SCREEN 求余
            return items.size() % NUMBER_PER_SCREEN;
            }
     //否则每屏显示的程序数量为 NUMBER_PER_SCREEN
            return NUMBER_PER_SCREEN;
        }
        @Override
        public Object getItem(int position) {
        //TODO 自动存根法
        //根据 screenNo 计算第 position 个列表项的数据
            return items.get(screenNo * NUMBER_PER_SCREEN + position);
```

```
        }
        @Override
        public long getItemId(int position) {
            //TODO 自动存根法
            return position;
        }
        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
            //TODO 自动存根法
            View view = convertView;
            if(convertView == null)
            {
                //加载 R.layout.labelicon 布局文件
                view = inflater.inflate(R.layout.lablicon, null);
            }
            //获取 R.layout.labelicon 布局文件中的 ImageView 组件,并设置图标
            ImageView imageView = (ImageView)view.findViewById(R.id.imageview);
            imageView.setImageDrawable(((DataItem)getItem(position)).drawable);
        //获取 R.layout.labelicon 布局文件中的 TextView 组件,并设置文本
            TextView textView = (TextView)view.findViewById(R.id.textview);
            textView.setText(((DataItem)getItem(position)).dataName);
            return view;
        }
    };
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

(4) 在 res\layout 目录下创建一个 labelicon.xml 文件。BaseAdapter 的 getView()只是简单加载了 labelicon 布局文件,并使用当前列表项的图片数据填充 labelicon 布局文件中的 ImageView,使用当前列表项的文本数据填充 labelicon 布局文件中的 TextView,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:gravity = "center"
    android:background = "#aabbcc">
    <ImageView
        android:id = "@ + id/imageview"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content" />
    <TextView
        android:id = "@ + id/textview"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:gravity = "center"/>
</LinearLayout>
```

(5) 在 res\layout 目录下创建一个 slidelistview.xml 文件,用于实现黑体字使用,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< GridView xmlns:android = "http://schemas.android.com/apk/res/android"
    android:numColumns = "4"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent" >
</GridView >
```

(6) 在 res 文件夹下创建一个 anim 文件夹,在文件中创建 4 个布局文件。实现当用户单击">"按钮时,程序的事件处理函数将会控制 ViewSwitcher 调用 showNext()方法显示下一屏的程序列表,此时 screenNo 被加 1,因而 Adapter 将会动态计算下一屏的程序列表,再将该 Adapter 传给 ViewSwitcher 要显示的 GridView。为了实现 ViewSwitcher 切换 View 时的动画效果,程序的事件处理方法中调用了 ViewSwitcher 的 setAnimation()和 setOutAnimation()方法来设置动画效果。本程序不仅利用了 Android 系统提供的两个动画资源,还自行提供了动画资源。

① slide_in_right 动画资源的代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android" >
    <!-- 设置从右边拖进来的动画
    android:duration 指定动画持续时间 -->
< translate
    android:fromXDelta = "100 % p"
    android:toXDelta = "0"
    android:duration = "@android:integer/config_mediumAnimTime"/>
</set >
```

② slide_out_left 动画资源的代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android">
    <!-- 设置从左边拖出去的动画
      android:duration 指定动画持续时间 -->
      < translate android:fromXDelta = "0"
          android:toXDelta = " - 100 % p"
          android:duration = "@android:integer/config_mediumAnimTime"/>
</set >
```

③ slide_in_left 动画资源的代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android" >
    <!-- 设置从左边拖进来的动画
    android:duration 指定动画持续时间 -->
< translate
    android:fromXDelta = " - 50 % p"
    android:toXDelta = "0"
    android:duration = "@android:integer/config_mediumAnimTime"/>
</set >
```

④ slide_out_right 动画资源的代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android">
    <!-- 设置从左边拖出去的动画
      android:duration 指定动画持续时间 -->
      < translate
```

```
            android:fromXDelta = "0"
            android:toXDelta = "50 % p"
            android:duration = "@android:integer/config_mediumAnimTime"/>
</set>
```

运行程序,默认效果如图 5-7 所示。当单击下侧的向左向右拖动按钮时,即可实现翻页。



图 5-7 图像的翻页

5.3.2 ViewFlipper 组件

当 ScrollView 失败时,找到了 ViewFlipper 控件,此控件可以左右滑动,而且可以和 ListView 控件结合来达到上下、左右滑动的效果。用到了 ViewFlipper 控件、Animation 动画、手势类 GestureDetector。

ViewFlipper 是 Android 官方提供的一个 View 容器类,继承 ViewAnimator 类,用于实现页面切换,也可以设定时间间隔,让它自动播放。

ViewAnimator 继承 FrameLayout,所以 ViewFlipper 的 Layout 里面可以放置多个 View,继承关系如图 5-8 所示。

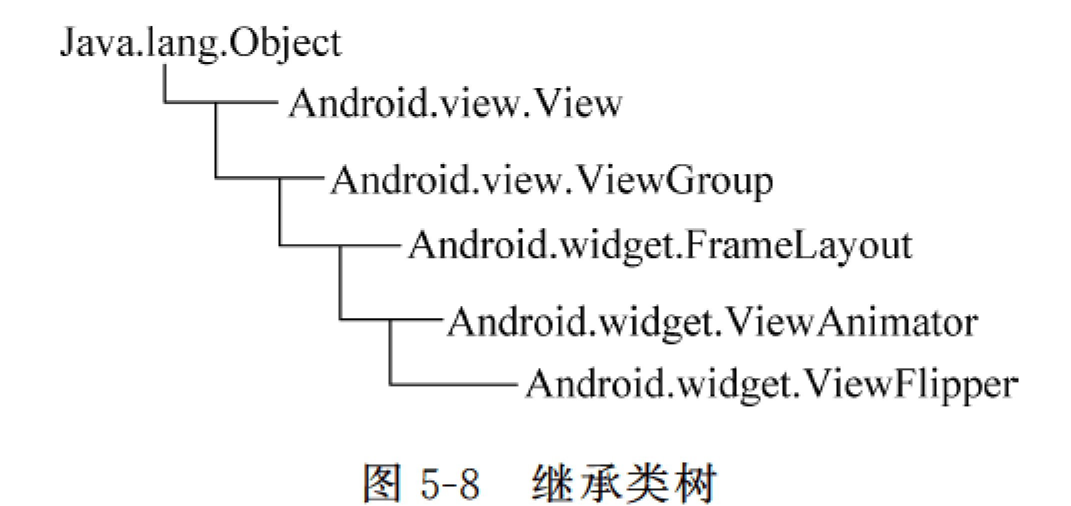


图 5-8 继承类树

下面通过一个案例来演示 ViewFlipper 组件的用法。

【例 5-9】 利用 ViewFlipper 组件图像的切换。其具体操作步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li5_8ViewFlipper。

(2) 打开 res\layout 目录下的 main.xml 布局文件,在布局文件中声明两个 Button 控件及一个 ViewFlipper 组件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
    < LinearLayout
        android:orientation = "horizontal"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:gravity = "center">
    < Button
        android:id = "@ + id/btnPrevious"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_marginRight = "5dip"
        android:text = "上一个" />
    < Button
        android:id = "@ + id/btnNext"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_marginLeft = "5dip"
        android:text = "下一个"/>
</LinearLayout >
< ViewFlipper
    android:id = "@ + id/flipper"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:gravity = "center"
    android:background = "#aabbcc">
    </ViewFlipper >
</LinearLayout >
```

(3) 打开 src\fs.li5_8viewflipper 包下的 MainActivity.java 文件,在文件中实现文字及图像的相互切换等功能,其代码为:

```
package fs.li5_8viewflipper;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup.LayoutParams;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.ViewFlipper;
public class MainActivity extends Activity {
    private Button previous, next;
    private ViewFlipper flipper;
    /** 第一次调用活动. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        initiaView();
        flipper.addView(addButtonByText("开始"),new LayoutParams(LayoutParams.FILL_PARENT,
LayoutParams.WRAP_CONTENT));
        flipper.addView(addTextByText("ViewFlipper 组件"));
        flipper.addView(addImageById(R.drawable.b1));
        flipper.addView(addTextByText("演示"));
        flipper.addView(addImageById(R.drawable.b4)); flipper.addView(addButtonByText("结
束"), new LayoutParams(LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT));
        }
         /** 初始化视图 */
    private void initiaView(){
        previous = (Button) findViewById(R.id.btnPrevious);
        next = (Button) findViewById(R.id.btnNext);
        flipper = (ViewFlipper) findViewById(R.id.flipper);
    flipper.setInAnimation(AnimationUtils.loadAnimation(this,android.R.anim.fade_in));
    flipper.setOutAnimation(AnimationUtils.loadAnimation(this,android.R.anim.fade_out));
        previous.setOnClickListener(listener);
        next.setOnClickListener(listener);
        }
    private OnClickListener listener = new OnClickListener(){
        public void onClick(View v) {
            //TODO 自动存根法
            switch(v.getId()){
            case R.id.btnPrevious:
                flipper.showPrevious();
                break;
                case R.id.btnNext:
                    flipper.showNext();
                    break;
                    }
            }
        };
        public View addTextByText(String text){
            TextView tv = new TextView(this);
            tv.setText(text);
            tv.setGravity(1);
            return tv;
            }
        public View addImageById(int id){
            ImageView iv = new ImageView(this);
            iv.setImageResource(id);
            return iv;
            }
        public View addButtonByText(String text){
            Button btn = new Button(this);
            btn.setText(text);
            return btn;
        }
}
```

运行程序,默认界面如图 5-9(a)所示,当单击界面中的“开始”按钮时,即弹出相应的文字,

当单击“上一个”或“下一个”按钮时，即弹出相应的图片，如图 5-9(b)和图 5-9(c)所示。



(a) 默认界面　(b) 显示图片　(c) 显示文字

图 5-9　ViewFlipper 组件实现图像切换

5.4　图 像 扭 曲

在 Android 中提供了 drawBitmapMesh 类实现了图像的扭曲 Canvas 的 drawBitmapMesh 定义如下：

```
public void drawBitmapMesh(Bitmap bitmap, int meshWidth, int meshHeight, float [] verts, int
vertOffset, int [] colors, int colorOffset, Paint paint)
```

其用于表示将图像绘制在网格上，简言之，可以将画板想象成一张格子布，在这张布上绘制图像。对于一个网格端点均匀分布的网格来说，横向有 meshWidth+1 个顶点，纵向有 meshHeight+1 个端点。顶点数组 verts 是以行优先的数组(二维数组以一维数组表示，先行后列)。网格可以不均匀分布，参数定义如下：

- Bitmap：需要绘制在网格上的图像。
- meshWidth：网格的宽度方向的数目(列数)，为 0 时不绘制图像。
- meshHeight：网格的高度方向的数目(行数)，为 0 时不绘制图像。
- verts：为(x，y)对的数组，表示网格顶点的坐标，至少需要有(meshWidth+1)×(meshHeight+1)×2+meshOffset 个(x，y)坐标。
- vertOffset：用于控制 verts 数组中开始跳过的(x，y)对的数目。
- Colors：可以为空，不为空时为每个顶点定义对应的颜色值，至少需要有(meshWidth+1)×(meshHeight+1)×2+meshOffset 个(x，y)坐标。
- colorOffset：colors 数组中开始跳过(x，y)对的数目。
- paint：可以为空。

值得注意的是，当程序希望调用 drawBitmapMesh 方法对位图进行扭曲时，关键是计算 verts 数组的值——该数组的值记录了扭曲后的位图上各“顶点”的坐标。

下面案例来演示 drawBitmapMesh 控件的用法。

【例 5-10】 利用 drawBitmapMesh 控件图像的扭曲。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li5_9drawBitmapMesh。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在布局文件中只定义一个 RelativeLayout 布局，并设置屏幕的背景图，其代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc">
</RelativeLayout>
```

(3) 打开 src\fs.li5_9drawbitmapmesh 包下的 MainActivity.java 文件，在文件中实现当单击图像某处时，对某处进行扭曲操作，其代码为：

```
package fs.li5_9drawbitmapmesh;
import android.app.Activity;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.os.Bundle;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.View;
public class MainActivity extends Activity {
    /** 第一次调用活动. */
    private Bitmap bitmap;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new MyView(this,R.drawable.fj));
    }
    private class MyView extends View
    {
        //定义两个常量,这两个常量指定该图片的横向和纵向都被划分为 20 格
        private final int WIDTH = 20;
        private final int HEIGHT = 20;
        //记录该图片上包含 441 个顶点
        private final int COUNT = (WIDTH + 1) * (HEIGHT + 1);
        //定义一个数组,记录 Bitmap 上的 21×21 个点的坐标
        private final float[] verts = new float[COUNT * 2];
        //定义一个数组,记录图片上的 21×21 个点经过扭曲后的坐标
        //对图片扭曲的关键就是修改该数组里元素的值
        private final float[] orig = new float[COUNT * 2];
        public MyView(Context context, int drawableId) {
            super(context);
            setFocusable(true);
```

```
        //根据指定资源加载图片
        bitmap = BitmapFactory.decodeResource(getResources(), drawableId);
        //获取图片宽度和高度
        float bitmapWidth = bitmap.getWidth();
        float bitmapHeight = bitmap.getHeight();
        int index = 0;
        for(int y = 0; y <= HEIGHT; y++)
        {
            float fy = bitmapHeight * y / HEIGHT;
            for(int x = 0 ; x <= WIDTH; x++)
            {
                float fx = bitmapWidth * x / WIDTH;
              //初始化 orig、verts 数组
              //初始化,orig、verts 两个数组均匀地保存了 21×21 个点的 x,y 坐标
                orig[index * 2 + 0] = verts[index * 2 + 0] = fx;
                orig[index * 2 + 1] = verts[index * 2 + 1] = fy;
                index += 1;
            }
        }
        //设置背景色
        setBackgroundColor(Color.WHITE);
    }
    protected void onDraw(Canvas canvas)
    {
        //对图片按 verts 数组进行扭曲
        //从第一个点(由第 5 个参数 0 控制)开始扭曲
        canvas.drawBitmapMesh(bitmap, WIDTH, HEIGHT, verts, 0, null, 0, null);
    }
    //工具方法,用于根据触摸事件的位置计算 verts 数组里各元素的值
    private void warp(float cx, float cy)
    {
        for(int i = 0; i < COUNT * 2; i += 2)
        {
            float dx = cx - orig[i + 0];
            float dy = cy - orig[i + 1];
            float dd = dx * dx + dy * dy;
            //计算每个坐标点与当前点(cx,cy)之间的距离
            float d = (float)Math.sqrt(dd);
            //计算扭曲度,距离当前点(cx,cy)越远,扭曲度越小
            float pull = 80000 / ((float)(dd * d));
        //对 verts 数组(保存图片上 21×21 个点经过扭曲后的坐标)重新赋值
            if(pull >= 1)
            {
                verts[i + 0] = cx;
                verts[i + 1] = cy;
            }
            else
            {
                //控制各顶点向触摸事件发生点偏移
                verts[i + 0] = orig[i + 0] + dx * pull;
                verts[i + 1] = orig[i + 1] + dx * pull;
            }
        }
        //通知 View 组件重绘
        invalidate();
```

```
        }
        public boolean onTouchEvent(MotionEvent event)
        {
            //调用 warp 方法根据触摸屏事件的坐标点来扭曲 verts 数组
            warp(event.getX(), event.getY());
            return true;
        }
    }
}
```

运行程序,默认界面如图 5-10 所示,当单击图像某处时,效果如图 5-10(b)和图 5-10(c)所示。



(a) 默认界面　　(b) 图像扭曲1　　(c) 图像扭曲2

图 5-10　drawBitmapMesh 控件用法

5.5　动画集合类

AnimationSet 类是 Android 系统中的动画集合类,用于控制 View 对象进行多个动作的组合,该类继承 Animation 类。AnimationSet 类中的很多方法都与 Animation 类一致。如果需要对一个控件进行多种动画设置,可以采用 animationset,其方法有:

(1) 为动画集合对象添加动画对象,其代码为:

```
Public void addAnimation(Animation a)
```

(2) 创建一个 animationset,其代码为:

```
AnimationSet animationset = new AnimationSet(true);
```

(3) 创建一个 alphaanimation,其代码为:

```
Animation alphaanimation = AnimationUtils.loadAnimation(
Animation1Activity.this,R.anim.alpha);
ScaleAnimation scaleanimation = new ScaleAnimation(
1f, 0f, 1f, 0f, Animation.RELATIVE_TO_SELF, Animation.RELATIVE_TO_SELF);
```

(4) 设置动画时间,其代码为:

```
animationset.addAnimation(alphaanimation);
animationset.addAnimation(scaleanimation);
animationset.setDuration(2000);
animationset.setStartOffset(2000);
imageview.setAnimation(animationset);
```

下面通过案例来演示设置一个组合动画效果。

【例 5-11】 利用 AnimationSet 组件来演示图像的移动效果。其具体操作实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li5_10AnimationSet 的工程。

(2) 编写布局文件，布局一个文本框、两个按钮及一个图片视图控件。打开 res\Layout 目录下的 main.xml 文件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:background = "@drawable/h1"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent" android:id = "@ + id/LL">
<TextView
    android:layout_height = "wrap_content"
    android:layout_width = "match_parent"
    android:id = "@ + id/textView1"
    android:text = "动画集合效果"/>
<Button
    android:layout_height = "wrap_content"
    android:layout_width = "match_parent"
    android:id = "@ + id/button1"
    android:text = "开始"/>
<Button
    android:layout_height = "wrap_content"
    android:layout_width = "match_parent"
    android:id = "@ + id/button2"
    android:text = "结束"/>
<ImageView
    android:src = "@drawable/pig"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:id = "@ + id/imageView1"/>
</LinearLayout>
```

(3) 编写 Activity 文件。打开 src/com.example.animation_set 包下的 MainActivity.java 文件，其代码为：

```
package fs.li5_10animationset;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.animation.AlphaAnimation;
import android.view.animation.Animation;
import android.view.animation.AnimationSet;
import android.view.animation.RotateAnimation;
import android.view.animation.ScaleAnimation;
import android.view.animation.TranslateAnimation;
import android.widget.Button;
```

```
import android.widget.ImageView;
import android.widget.TextView;
public class MainActivity extends Activity
{
    /** 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        //重载 onCreate 方法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final ImageView image = (ImageView)findViewById(R.id.imageView1);
        Button btn1 = (Button)findViewById(R.id.button1);
        Button btn2 = (Button)findViewById(R.id.button2);
        //位置变化动画
        final Animation translateAnimation = new TranslateAnimation(0,300,0,300);
                                                         //尺寸变化动画
        final Animation scaleAnimation = new ScaleAnimation(0f,1f,0f,1f,Animation.RELATIVE_TO_
SELF,0.5f,Animation.RELATIVE_TO_SELF,0.5f);
     final Animation alphaAnimation = new AlphaAnimation(0.1f,1.0f);          //透明度变化动画
        final AnimationSet set = new AnimationSet(true);       //创建动画集对象
        btn1.setOnClickListener(new View.OnClickListener()
        {
            //监听器的设置
            @Override
            public void onClick(View v)
            {
                //TODO 自动存根法
                translateAnimation.setDuration(2000);          //位置变化动画的持续时间
                scaleAnimation.setDuration(2000);              //尺寸变化动画的持续时间
                alphaAnimation.setDuration(2000);              //透明度渐变动画的持续时间
                set.addAnimation(translateAnimation);          //添加位置变化动画
                set.addAnimation(scaleAnimation);              //添加尺寸变化动画
                set.addAnimation(alphaAnimation);              //添加透明度渐变动画
                set.setFillAfter(true);                        //停留在最后的位置
                set.setFillEnabled(true);
                image.setAnimation(set);                       //动画
                set.startNow();                                //启动动画
            }
        });

btn2.setOnClickListener(new View.OnClickListener()
        {
            //设置监听器
            @Override
            public void onClick(View v)
            {
                //TODO 自动存根法
                set.cancel();                                  //取消动画执行
            }
        });
    }
}
```

在以上代码中，构造了位置变化、尺寸变化和透明度变化的动画对象。运行程序，初始界面如图 5-11(a)所示。当单击界面中的“开始”按钮时，即动画开始运动，当单击界面中“结束”按钮时，动画即停止，效果如图 5-11(b)所示。



(a) 初始界面



(b) 动画移动

图 5-11　AnimationSet 控件的用法

5.6　绘图容器

SurfaceView 由于可以直接从内存或 DMA 等硬件接口取得图像数据，因此是个非常重要的绘图容器。其他的特性是：可以在主线程之外的线程中向屏幕绘图。可以避免画图任务繁重时造成主线程阻塞，从而提高了程序的反应速度。在游戏开发中多用到 SurfaceView，游戏中的背景、人物、动画等尽量在画布 canvas 中画出。

SurfaceView 的核心在于提供了两个线程：UI 线程和渲染线程。这里应注意：

(1) 所有 SurfaceView 和 SurfaceHolder.Callback 的方法都应该在 UI 线程里调用，一般来说就是应用程序主线程。渲染线程所要访问的各种变量应该作同步处理。

(2) 由于 Surface 可能被销毁，它只在 SurfaceHolder.Callback.surfaceCreated() 和 SurfaceHolder.Callback.surfaceDestroyed()之间有效，所以要确保渲染线程访问的是合法有效的 Surface。

1. SurfaceView 类实现

首先继承 SurfaceView 并实现 SurfaceHolder.Callback 接口。

使用接口的原因，因为使用 SurfaceView 有一个原则，所有的绘图工作必须得在 Surface 被创建之后才能开始(Surface 是表面，这个概念在图形编程中常常被提到。基本上可以把它当作显存的一个映射，写入到 Surface 的内容可以被直接复制到显存从而显示出来，使得显示速度非常快)，而在 Surface 被销毁之前必须结束。所以 Callback 中的 surfaceCreated 和 surfaceDestroyed 就成了绘图处理代码的边界。

需要重写的方法有：

(1) public void surfaceChanged(SurfaceHolder holder,int format,int width,int height){}：在 surface 的大小发生改变时激发。

(2) public void surfaceCreated(SurfaceHolder holder){}：在创建时激发，一般用来调用

画图的线程。

(3) public void surfaceDestroyed(SurfaceHolder holder) {}:销毁时激发,一般用来将画图的线程停止、释放。

其过程为:首先继承 SurfaceView 并实现 SurfaceHolder. Callback 接口;接着 SurfaceView. getHolder()获得 SurfaceHolder 对象;然后为 SurfaceHolder. addCallback(callback)添加回调函数;用 SurfaceHolder. lockCanvas()获得 Canvas 对象并锁定画布;用 Canvas 绘画;最后应用 SurfaceHolder. unlockCanvasAndPost(Canvas canvas)结束锁定画图,并提交改变,将图形显示。

2. SurfaceHolder 类概述

这里用到了一个类 SurfaceHolder,可以把它当成 Surface 的控制器,用来操纵 Surface。处理它的 Canvas 上画的效果和动画,控制表面、大小、像素等。

几个需要注意的方法:

(1) abstract void addCallback(SurfaceHolder. Callback callback):给 SurfaceView 当前的持有者一个回调对象。

(2) abstract Canvas lockCanvas():锁定画布,一般在锁定后就可以通过其返回的画布对象 Canvas,在其上面画图等。

(3) abstract Canvas lockCanvas(Rect dirty):锁定画布的某个区域进行画图等,因为画完图后,会调用下面的 unlockCanvasAndPost 来改变显示内容。对内存要求较高的游戏,可以不用重画 dirty 外的区域像素,这样可以提高速度。

(4) abstract void unlockCanvasAndPost(Canvas canvas):结束锁定画图,并提交改变。

3. SurfaceView 双缓冲应用

双缓冲是为了防止动画闪烁而实现的一种多线程应用,基于 SurfaceView 的双缓冲实现很简单,开一条线程并在其中绘图即可。

下面通过一个案例来演示 SurfaceView 的用法。

【例 5-12】 利用 SurfaceView 控件实现双缓冲,以及介绍类似、更高效的实现方法。其主要实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li5_11SurfaceView。

(2) 打开 res\layout 目录下的 main. xml 布局文件,在文件中声明两个 Button 控件及一个 SurfaceView 控件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
     android:layout_width = "fill_parent"
     android:layout_height = "fill_parent"
     android:orientation = "vertical"
     android:background = "#aabbcc">
<LinearLayout
     android:id = "@ + id/LinearLayout1"
     android:layout_width = "wrap_content"
     android:layout_height = "wrap_content">
<Button
     android:id = "@ + id/Button1"
     android:layout_width = "wrap_content"
     android:layout_height = "wrap_content"
     android:text = "单个独立线程"/>
<Button
     android:id = "@ + id/Button2"
```

```
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "两个独立线程"/>
  </LinearLayout >
  < SurfaceView
        android:id = "@ + id/SurfaceView1"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"/>
</LinearLayout >
```

(3) 打开 src\fs. li5_11surfaceview 包下的 MainActivity. java 文件,在文件中实现双缓冲,其代码为:

```
package fs.li5_11surfaceview;
import java.lang.reflect.Field;
import java.util.ArrayList;
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Rect;
import android.os.Bundle;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.widget.Button;
public class MainActivity extends Activity {
    /** 第一次调用活动. */
    Button btnSingleThread, btnDoubleThread;
    SurfaceView sfv;
    SurfaceHolder sfh;
    ArrayList < Integer > imgList = new ArrayList < Integer >();
    int imgWidth, imgHeight;
    Bitmap bitmap;                                    //独立线程读取,独立线程绘图
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btnSingleThread = (Button) this.findViewById(R.id.Button1);
        btnDoubleThread = (Button) this.findViewById(R.id.Button2);
        btnSingleThread.setOnClickListener(new ClickEvent());
        btnDoubleThread.setOnClickListener(new ClickEvent());
        sfv = (SurfaceView) this.findViewById(R.id.SurfaceView1);
        sfh = sfv.getHolder();
        sfh.addCallback(new MyCallBack());      //自动运行 surfaceCreated 以及 surfaceChanged
    }
    class ClickEvent implements View.OnClickListener {
        @Override
        public void onClick(View v) {
            if (v == btnSingleThread) {
                new Load_DrawImage(0, 0).start();          //开一条线程读取并绘图
            } else if (v == btnDoubleThread) {
                new LoadImage().start();                   //开一条线程读取
                new DrawImage(imgWidth + 10, 0).start();   //开一条线程绘图
            }
        }
```

```
        }
    class MyCallBack implements SurfaceHolder.Callback {
        @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
            Log.i("Surface:", "Change");
            }
        @Override
        public void surfaceCreated(SurfaceHolder holder) {
            Log.i("Surface:", "Create");
            //用反射机制来获取资源中的图片 ID 和尺寸
            Field[] fields = R.drawable.class.getDeclaredFields();
            for (Field field : fields) {
                if (!"icon".equals(field.getName()))      //除了 icon 之外的图片
                    {
                    int index = 0;
                    try {
                        index = field.getInt(R.drawable.class);
                        } catch (IllegalArgumentException e) {
                            //TODO 自动存根法
                            e.printStackTrace();
                            } catch (IllegalAccessException e) {
                                //TODO 自动存根法
                                e.printStackTrace();
                            }
                    //保存图片 ID
                    imgList.add(index);
                    }
                }
            //取得图像大小
    Bitmap bmImg = BitmapFactory.decodeResource(getResources(),imgList.get(0));
            imgWidth = bmImg.getWidth();
            imgHeight = bmImg.getHeight();
            }
        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
            Log.i("Surface:", "Destroy");
            }
        }
    /** 读取并显示图片的线程 */
    class Load_DrawImage extends Thread {
        int x, y;
        int imgIndex = 0;
        public Load_DrawImage(int x, int y) {
            this.x = x;
            this.y = y;
            }
        public void run() {
            while (true) {
Canvas c = sfh.lockCanvas(new Rect(this.x, this.y, this.x + imgWidth, this.y + imgHeight));
Bitmap bmImg = BitmapFactory.decodeResource(getResources(),imgList.get(imgIndex));
                c.drawBitmap(bmImg, this.x, this.y, new Paint());
                imgIndex++;
                if (imgIndex == imgList.size())
                    imgIndex = 0;
                sfh.unlockCanvasAndPost(c);            //更新屏幕显示内容
             }
        }
     };
```

```
    /* 只负责绘图的线程 */
    class DrawImage extends Thread {
        int x, y;
        public DrawImage(int x, int y) {
            this.x = x;
            this.y = y;
            }
        public void run() {
            while (true) {
                if (bitmap!= null) {                    //如果图像有效
Canvas c = sfh.lockCanvas(new Rect(this.x, this.y, this.x + imgWidth, this.y + imgHeight));
                    c.drawBitmap(bitmap, this.x, this.y, new Paint());
                    sfh.unlockCanvasAndPost(c);         //更新屏幕显示内容
                }
            }
        }
    };
   /* 只负责读取图片的线程 */
   class LoadImage extends Thread {
       int imgIndex = 0;
       public void run() {
           while (true) {
               bitmap = BitmapFactory.decodeResource(getResources(), imgList.get(imgIndex));
               imgIndex++;
               if (imgIndex == imgList.size())          //如果到尽头则重新读取
                   imgIndex = 0;
               }
           }
   };
}
```

运行程序，默认界面如图 5-12(a)所示；当单击界面中的“单个独立线程”按钮，效果如图 5-12(b)所示；单击界面中的“两个独立线程”按钮时，效果如图 5-12(c)所示。



(a) 默认界面　(b) 单个独立线程界面　(c) 两个独立线程界面

图 5-12　SurfaceView 控件用法

5.7 消息提示

当程序有大量消息、图片需要向用户提示时，可以考虑使用前面介绍的对话框，但如果程序只有少量信息要向用户呈现，则可以考虑使用“更轻量级”的对话框，即 Android 提供的消息提示框。

5.7.1 Toast 控件

Android 中提供一种简单的 Toast 消息提示框机制，可以在用户单击某些按钮后，提示用户一些信息，提示的信息不能被用户点击，Toast 的提示信息根据用户设置的显示时间后自动消失。Toast 的提示信息可以在调试程序时方便地显示某些想显示的内容。

在 Android 中可用两种方法创建 Toast，分别为：

- makeText(Context context, int resId, int duration)：参数 context 是 Toast 显示在哪个上下文，通常是当前 Activity；resId 指显示内容引用 Resource 那条数据，就是从 R 类中指定显示的消息内容；duration 指定显示时间；Toast 默认有 LENGTH_SHORT 和 LENGTH_LONG 两常量，分别表示短时间显示和长时间显示。
- makeText(Context context, CharSequence text, int duration)：参数 context 和 duration 与第一个方法相同，参数 text 可以自己写消息内容。

用上面任意方法创建 Toast 对象之后调用方法 show()即可显示。

例如：

```
Toast toast = Toast.makeText(ToastDemoActivity.this, "这是一个普通的 Toast!", Toast.LENGTH_
SHORT);
toast.show();
```

也可以通过以下两种方法来设置 Toast 显示位置。

- setGravity(int gravity, int xOffset, int yOffset)：三个参数分别表示(起点位置，水平向右位移，垂直向下位移)。
- setMargin(float horizontalMargin, float verticalMargin)：以横向和纵向的百分比设置显示位置，参数均为 float 类型(水平位移右正左负，竖直位移上正下负)。

例如，设置 Toast 显示位置(起点位置，水平向右位移，垂直向下位移)的 Java 代码为：

```
toast.setGravity(Gravity.TOP | Gravity.LEFT, 0, 200);
```

Toast 显示位置，以横向和纵向的百分比计算，参数均为 float 类型。Java 代码为：

```
toast.setMargin( - 0.5f, 0f);
```

Toast 的功能和用法都比较简单，通常只能显示简单的文本提示；如果应用需要显示诸如图片、列表之类的复杂提示，一般建议使用对话框来完成；开发者也通过 Toast 来完成，Toast 提供了一个 setView()方法，该方法允许开发者自己定义 Toast 显示的内容。

自定义一个 Toast 也简单，同样需要创建一个 Toast 对象，然后实现相应的方法即可，下面通过一个案例来演示 Toast 的用法。

【例 5-13】 弹出消息 Toast 对象的使用自定义方式。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li5_12Toast。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在布局文件中声明两个 Button 控件

及一个 EditText 控件,其代码为:

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc">
<EditText
    android:id = "@ + id/et"
    android:singleLine = "true"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:layout_marginLeft = "0dip"/>
    <Button
        android:id = "@ + id/bt1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_below = "@ + id/bt2"
        android:layout_marginTop = "26dp"
        android:text = "纯文本方式 Toast" />
    <Button
        android:id = "@ + id/bt2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/bt1"
        android:layout_below = "@ + id/et"
        android:layout_marginTop = "18dp"
        android:text = "自定义方式 Toast" />
</RelativeLayout>
```

(3) 在 res\layout 目录下创建一个 mytoast.xml 文件,用于实现自定义方式 Toast 显示图片控件 ImageView,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
<ImageView
    android:id = "@ + id/iv"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:src = "@drawable/b9"
    android:layout_gravity = "center_vertical"/>
</LinearLayout>
```

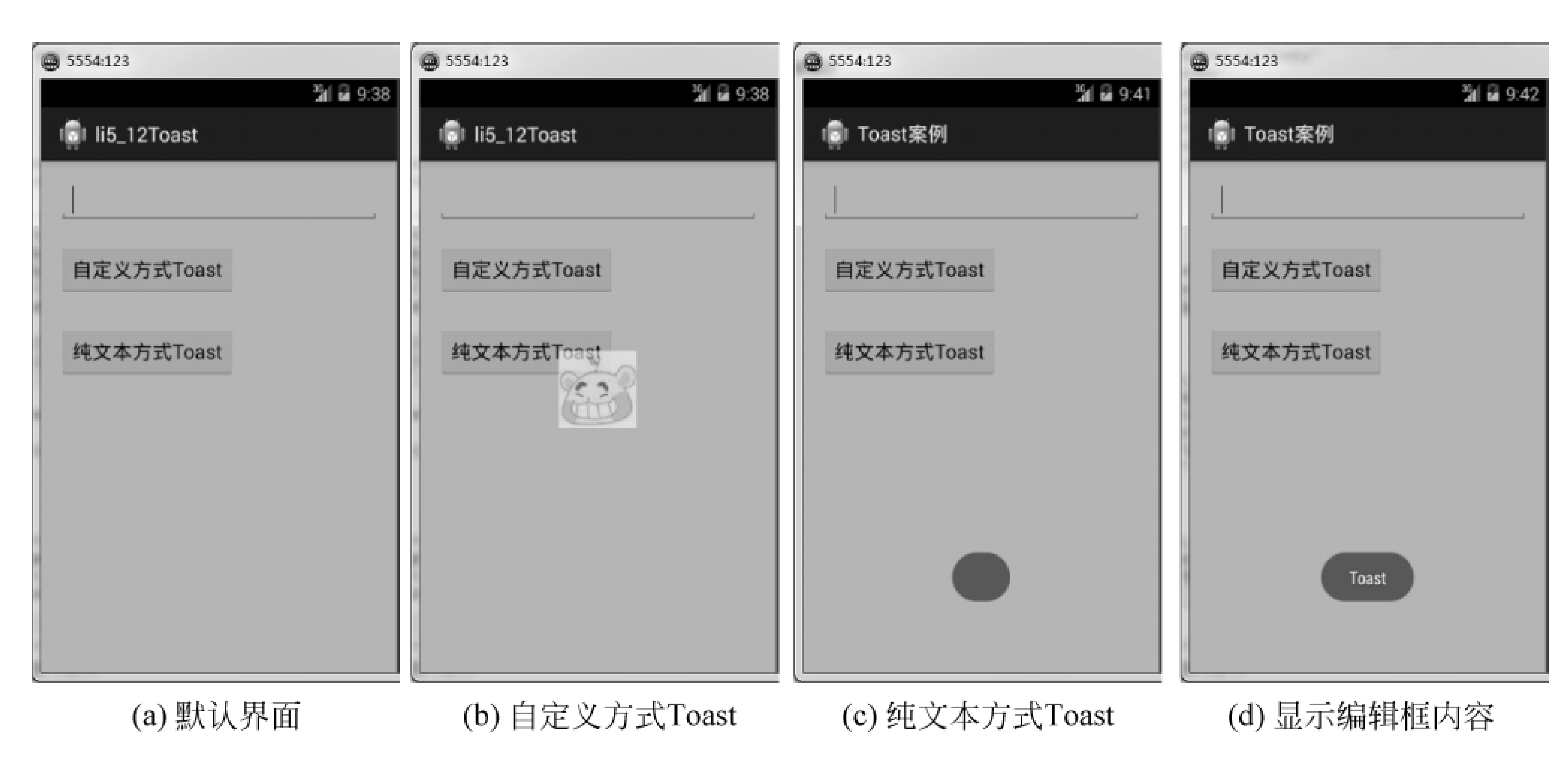
(4) 打开 src\fs.li5_12toast 包下的 MainActivity.java 文件,在文件中实现纯文本的 Toast 的使用以及自定义 Toast 的显示内容和显示位置,其代码为:

```
package fs.li5_12toast;
import android.app.Activity;
import android.os.Bundle;
```

```java
import android.text.Editable;
import android.view.Gravity;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends Activity{
    //声明两个 Button 对象
    private Button mybtn1,mybtn2;
    //声明一个 EditText 对象
    private EditText myedittext;
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        //加载 main.xml 布局文件
        setContentView(R.layout.main);
        //以 findViewById()方法取得 Button 对象
        mybtn1 = (Button)findViewById(R.id.bt1);
        mybtn2 = (Button)findViewById(R.id.bt2);
        //以 findViewById()方法取得 EditText 对象
        myedittext = (EditText)findViewById(R.id.et);
        //给 Button 对象绑定单击监听事件
        mybtn1.setOnClickListener(listener);
        mybtn2.setOnClickListener(listener);
        }
    //监听事件
    private OnClickListener listener = new OnClickListener(){
        @Override
        public void onClick(View v){
            switch (v.getId()){
            case R.id.bt1:
                //声明字符串变量
                Editable str;
                //得到由用户输入 EditText 的文字内容
                str = myedittext.getText();
                //使用 Toast.makeText()方法来产生 Toast 信息
    Toast.makeText(MainActivity.this, str.toString(), Toast.LENGTH_LONG).show();
                //清空 EditText
                myedittext.setText("");
                break;
                case R.id.bt2:
                    //导入布局文件
                    View view = getLayoutInflater().inflate(R.layout.mytoast, null);
                    //得到 Toast 对象
                    Toast toast = new Toast(MainActivity.this);
            //设置 Toast 对象的位置,三个参数分别为位置、X 轴偏移、Y 轴偏移
                    toast.setGravity(Gravity.CENTER, 0, 0);
                    //设置 Toast 对象的显示时间
                    toast.setDuration(Toast.LENGTH_LONG);
                    //设置 Toast 对象所要展示的视图
                    toast.setView(view);
                    //显示 Toast
                    toast.show();
                    break;
                    default:
                        break;
                    }
```

```
            }
        };
    }
```

运行程序，默认界面如图 5-13(a)所示；当单击界面中的“自定义方式 Toast”按钮时，效果如图 5-13(b)所示；当单击界面中的“纯文本方式 Toast”按钮时，效果如图 5-13(c)所示；当在编辑框中输入相应的字串，再单击“纯文本方式 Toast”按钮时，效果如图 5-13(d)所示。



(a) 默认界面　(b) 自定义方式Toast　(c) 纯文本方式Toast　(d) 显示编辑框内容

图 5-13　Toast 的使用

5.7.2　Notification 控件

当用户有没有接到的电话的时候，Android 顶部状态栏里就会出现一个小图标。提示用户有没有处理的快讯，当拖动状态栏时，可以查看这些快讯。Android 提供了 NotificationManager 来管理这个状态栏。

如果要添加一个 Notification，可以按照以下步骤完成。

(1) 获取 NotificationManager。

```
NotificationManager m_NotificationManager = (NotificationManager)this.getSystemService(NOTIFICATION_
SERVICE);
```

(2) 定义一个 Notification。

```
Notification m_Notification = new Notification();
```

(3) 设置 Notification 的各种属性。

① 设置通知在状态栏显示的图标。

```
m_Notification.icon = R.drawable.icon;
```

② 当点击通知时显示的内容。

```
m_Notification.tickerText = "Button1 通知内容……";
```

③ 通知时发出的默认声音。

```
m_Notification.defaults = Notification.DEFAULT_SOUND;
```

④ 设置通知显示的参数。

```
Intent m_Intent = new Intent(NotificationDemo.this,DesActivity.class);
PendingIntent m_PendingIntent = PendingIntent.getActivity(NotificationDemo.this, 0, m_Intent, 0);
```

```
m_Notification.setLatestEventInfo(NotificationDemo.this, "Button1", "Button1 通知", m_PendingIntent);
```

⑤ 可以理解为开始执行这个通知。

```
m_NotificationManager.notify(0,m_Notification);
```

(4) 既然可增加,即同样也可删除。当然是只是删除自己增加的内容。

```
m_NotificationManager.cancel(0);
```

此处的 0 是一个 ID 号码,和 notify 第一个参数 0 一样。

下面通过一个案例来演示 Notification 控件的用法。

【例 5-14】 在 Android 的 Notification 中显示进度条。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li5_13Notification。

(2) 打开 res\layout 目录下的 main.xml 布局文件,在文件中声明一个 TextView 控件和一个 Button 控件,其代码为:

```
<LinearLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "horizontal"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:padding = "10dp"
    android:background = "#aabbcc">
    <TextView
        android:id = "@+id/textView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "Android"
        android:textSize = "24dip"/>
    <Button
        android:id = "@+id/bt"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_marginRight = "14dp"
        android:text = "取消" />
</LinearLayout>
```

(3) 在 res\layout 目录下创建一个 dialog.xml 文件,在文件中声明一个 ImageView 控件、一个 ProgressBar 控件及一个 TextView 控件,其代码为:

```
<?xml version = "1.0" encoding = "utf-8"?>
<LinearLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "horizontal"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:padding = "10dp"
    android:background = "#aabbcc">
<ImageView
    android:id = "@+id/image"
    android:layout_width = "wrap_content"
    android:layout_height = "fill_parent" />
<ProgressBar
    android:id = "@+id/pb"
    android:layout_width = "180dip"
    android:layout_height = "wrap_content"
```

```
        style = "?android:attr/progressBarStyleHorizontal"
        android:layout_gravity = "center_vertical"/>
    < TextView
        android:id = "@ + id/tv"
        android:layout_width = "wrap_content"
        android:layout_height = "fill_parent"
        android:textSize = "16px"
        android:textColor = " # FF0000"/>
    </LinearLayout >
```

(4) 打开 src\fs. li5_13notification 包下的 MainActivity. java 文件,实现 Notification 提示框及进度条下载,其代码为:

```
package fs. li5_13notification;
import android. annotation. SuppressLint;
import android. app. Activity;
import android. app. Notification;
import android. app. NotificationManager;
import android. app. PendingIntent;
import android. content. Intent;
import android. os. Bundle;
import android. os. Handler;
import android. os. Message;
import android. service. notification. NotificationListenerService;
import android. view. View;
import android. widget. Button;
import android. widget. RemoteViews;
public class MainActivity extends Activity {
    //当前进度条里的进度值
    private int progress = 0;
    private RemoteViews view = null;
    private Notification notification = new Notification();
    private NotificationManager manager = null;
    private Intent intent = null;
    private PendingIntent pIntent = null;                        //更新显示
    private Handler handler = new Handler(){
        @Override
        public void handleMessage(Message msg) {
            //TODO 自动存根法
            view. setProgressBar(R. id. pb, 100, progress, false);
            //关键部分,如果不重新更新通知,进度条是不会更新的
            view. setTextViewText(R. id. tv, "下载" + progress + " % ");
            notification. contentView = view;
            notification. contentIntent = pIntent;
            manager. notify(0, notification);
            super. handleMessage(msg);
            }
        };
        @SuppressLint("NewApi")
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super. onCreate(savedInstanceState);
             setContentView (R. layout. main); manager = (NotificationManager) getSystemService
(NOTIFICATION_SERVICE);
            view = new RemoteViews(getPackageName(),R. layout. dialog);
            intent = new Intent(MainActivity. this,NotificationListenerService. class);
            pIntent = PendingIntent. getService(MainActivity. this, 0, intent, 0);
            Button button = (Button)findViewById(R. id. bt);
```

```
            button.setOnClickListener(new Button.OnClickListener(){
                @Override
                public void onClick(View v) {
                //通知的图标必须设置(其他属性为可选设置),否则通知无法显示
                    notification.icon = R.drawable.g1;
view.setImageViewResource(R.id.image, R.drawable.g1);        //启动一个线程用来更新 progress
                    new Thread(new Runnable(){
                        @Override
                        public void run() {
                            for(int i = 0;i < 20;i++){
                                progress = (i + 1) * 5;
                                try {
                                    if(i < 19){
                                        Thread.sleep(1000);
                                    }else {
                                        Thread.currentThread().interrupt();
                                        }
                                    } catch (InterruptedException e) {
                                        e.printStackTrace();
                                        }
                                Message msg = new Message();
                                handler.sendMessage(msg);
                                }
                            }
                        }).start();
                }
            });
        }
}
```

运行程序,默认效果如图 5-14(a)所示;当单击屏幕中的“取消”按钮时,效果如图 5-14(b)所示。



(a) Notification提示框　　(b) 进度条

图 5-14　Notification 控件用法

第 6 章 Android 对话框

在 Android 开发中，经常需要在 Android 界面上弹出一些对话框，如询问用户或让用户选择。这些功能称为 Android Dialog 对话框，下面分别介绍 9 种 Android Dialog 对话框的使用方法。

6.1 对话框概述

对话框是 Activity 运行时显示的小窗口，当显示对话框时，当前 Activity 失去焦点而由对话框负责所有的人机交互。一般来说，对话框用于提示消息或弹出一个与程序主进程直接相关的小程序。

对话框是作为 Activity 的一部分被创建和显示的，在程序中通过开发回调方法 onCreateDialog 来完成对话框的创建，该方法需要传入代表对话框的 id 参数。如果需要显示对话框，则调用 showDialog 方法传入对话框的 id 来显示指定的对话框。

当对话框第一次被显示时，Android 会调用 onCreateDialog 方法来创建对话框实例，之后将不再重复创建该实例，这点和被选菜单类似。同时，每次对话框在被显示之前都会调用 onPrepareDialog 方法，如果不重写该方法，那么每次显示的对话框将会是最初创建的那个。

当需要关闭对话框时，可以调用 Dialog 类的 dismiss 方法来实现，但是要注意的是以这种方式关闭的对话框并不会彻底消失，Android 会在后台保留其状态。如果需要让对话框在关闭之后彻底被清除，要调用 removeDialog 方法并传入 Dialog 的 id 值来彻底释放对话框。

提示：如果需要在调用 dismiss 方法关闭对话框时执行一些特定的工作，则可以为对话框设置 onDismissListener 并重写其中的 onDismiss 方法来开发特定的功能。

6.2 弹出式对话框

AlertDialog 对话框是常用的用于显示信息的方式，不能直接通过构造方法构建，而要由 AlertDialog. Builder 类来创建。AlertDialog 对话框的标题、按钮及按钮要响应的事件也由 AlertDialog. Builder 设置。

在使用 AlertDialog. Builder 创建对话框时常用的方法为：

- setTitle()：设置对话框标题。
- setIcon()：设置对话框图标。
- setMessage()：设置对话框的提示信息。
- setPositiveButton()：为对话框添加 yes 按钮。
- setNegativeButton()：为对话框添加 no 按钮。
- setNeutralButton()：为对话框添加第三个按钮。

6.2.1 简单对话框

使用简单的 Dialog 创建对话框按如下步骤进行：

- 创建 AlertDialog. Builder 对象，该对象为 AlertDialog 的创建器。
- 创建 AlertDialog. Builder 的方法为对话框设置图标、标题、内容等。

调用 AlertDialog. Builder 的 create()方法创建 AlertDialog 对话框。

调用 AlertDialog 的 show()方法显示对话框。

【例 6-1】 创建一个简单的 AlertDialog 并显示它。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 AlertDialog_1。

(2) 打开 src\fs. alertdialog_1 包下的 MainActivity. java 文件，在文件中实现简单对话框的显示，其代码为：

```
package fs.alertdialog_1;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.os.Bundle;
public class MainActivity extends Activity {
    /* 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Dialog alertDialog = new AlertDialog.Builder(this).
                setTitle("对话框的标题").
                setMessage("对话框的内容").
                setIcon(R.drawable.fb1).
                create();
        alertDialog.show();
    }
}
```

运行程序，效果如图 6-1 所示。



图 6-1 简单对话框

6.2.2 带按钮对话框

上面的例子代码简单，也容易实现，下面例子中将在这个 AlertDialog 上面添加几个按钮，用于实现对话框的相应操作。

【例 6-2】 创建一个带按钮的 AlertDialog，并显示它。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 AlertDialog_2。

(2) 打开 src \ fs. alertdialog _ 2 包下的 MainActivity. java 文件，在文件中为简单对话框添加“取消”、“查看详情”及“确定”按钮，其代码为：

```
package fs.alertdialog_2;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
```

```
import android.content.DialogInterface;
import android.os.Bundle;
public class MainActivity extends Activity {
    /** 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Dialog alertDialog = new AlertDialog.Builder(this).
                setTitle("确定删除?").
                setMessage("您确定删除该条信息吗?").
                setIcon(R.drawable.ic_launcher).
                setPositiveButton("确定", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        //TODO 自动存根法
                    }
                }).
                setNegativeButton("取消", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        //TODO 自动存根法
                    }
                }).
                setNeutralButton("查看详情", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        //TODO 自动存根法
                    }
                }).
                create();
        alertDialog.show();
    }
}
```

运行程序,效果如图 6-2 所示。

可以看到三个按钮添加到了 AlertDialog 上,三个没有添加事件处理的按钮,单击后只是关闭对话框,没有任何其他操作。

6.2.3 类似列表对话框

AlertDialog.Builder 除提供 setMessage 方法来设置对话框所显示的消息外,还提供如下方法来设置对话框显示列表内容。

- setItem(int itemsId,DialogInterface.OnClickListener listener):创建普通列表对话框。
- setMultiChoiceItems(CharSequence[] items,Boolean[] checkedItems,DialogInterface.OnMulti ChoiceClickListener listener):创建多选列表对话框。



图 6-2 带按钮的对话框

- setSingleChoiceItems (CharSequence[] items, int checkedItem, DialogInterface.OnClickListener listener):创建单选列表对话框。
- setAdapter(ListAdapter adapter,DialogInterface.OnClickListener listener):创建根据 ListAdapter 提供列表项的列表对话框。

【例 6-3】 创建一个类似于 ListView 控件的对话框。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 AlertDialog_List。

(2) 打开 src\fs.alertdialog_list 包下的 MainActivity.java 文件,在文件中实现 ListView 对话框的创建,并实现当选择对应的选项时,弹出对应的 Toast 消息,其代码为:

```
package fs.alertdialog_list;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.widget.Toast;
public class MainActivity extends Activity {
    /* 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
final String[] arrayFruit = new String[] { "战争片", "偶像片", "古装片", "青春校园片" };
        Dialog alertDialog = new AlertDialog.Builder(this).
                setTitle("你喜欢看哪类型电视?").
                setIcon(R.drawable.fb1)
                .setItems(arrayFruit, new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
Toast.makeText(MainActivity.this, arrayFruit[which], Toast.LENGTH_SHORT).show();
                    }
                }).
        setNegativeButton("取消", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        //TODO 自动存根法
                    }
                }).
                create();
        alertDialog.show();
    }
}
```

运行程序,效果如图 6-3 所示。



图 6-3 列表对话框

6.2.4 单选按钮对话框

用 setSingleChoiceItems(CharSequence[] items, int checkedItem, final OnClickListener listener)方法来实现类似 RadioButton 的 AlertDialog。

第一个参数是要显示的数据的数组,第二个参数是初始值(初始被选中的 item),第三个参数是单击某个 item 的触发事件。

【例 6-4】 创建 RadioButton 对话框并显示它。

在案例中设了一个 selectedFruitIndex 用来记住选中的 item 的 index，其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 AlertDialog_Radio。

(2) 打开 alertDialog_radio 包下的 MainActivity.java 文件，在文件中实现单选按钮对话框，其代码为：

```
package fs.alertdialog_radio;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.widget.Toast;
public class MainActivity extends Activity {
    private int selectedFruitIndex = 0;
    /* 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    final String[] arrayFruit = new String[] { "羽毛球", "篮球", "跳高", "游泳","长跑"};
        Dialog alertDialog = new AlertDialog.Builder(this).
                setTitle("你喜欢运动?").
                setIcon(R.drawable.fb1)
        .setSingleChoiceItems(arrayFruit, 0, new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        selectedFruitIndex = which;
                    }
                }).
                setPositiveButton("确认", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                          Toast.makeText(MainActivity.this, arrayFruit[selectedFruitIndex],
Toast.LENGTH_SHORT).show();
                    }
                }).
  setNegativeButton("取消", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        //TODO 自动存根法
                    }
                }).
                create();
        alertDialog.show();
    }
}
```

运行程序，效果如图 6-4 所示。

6.2.5 复选按钮对话框



图 6-4 单选按钮对话框

用 setMultiChoiceItems(CharSequence[] items, boolean[] checkedItems, final OnMultiChoiceClickListener listener)方法来实现类似 CheckBox 的 AlertDialog。

第一个参数是要显示的数据的数组，第二个参数是选中状态的数组，第三个参数是点击某个 item 的触发事件。

下面通过一个案例来创建 CheckBox 对话框并显示它。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 AlertDialog_Check。

(2) 打开 src\fs.alertdialog_check 包下的 MainActivity.java 文件，实现复选按钮对话框的选择，选择完成后单击“确定”按钮，即弹出 Toast 消息，其代码为：

```
package fs.alertdialog_check;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.widget.Toast;
public class MainActivity extends Activity
{
/* 第一次调用活动 */
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  final String[] arrayFruit = new String[] {"羽毛球", "篮球", "跳高", "游泳"};
  final boolean[] arrayFruitSelected = new boolean[] {true, true, false, false};
  Dialog alertDialog = new AlertDialog.Builder(this).
    setTitle("你喜欢的运动?").
    setIcon(R.drawable.fb1)
    .setMultiChoiceItems(arrayFruit, arrayFruitSelected, new
                          DialogInterface.OnMultiChoiceClickListener()
    {
     @Override
     public void onClick(DialogInterface dialog, int which, boolean isChecked)
     {
      arrayFruitSelected[which] = isChecked;
     }
    }).
    setPositiveButton("确认", new DialogInterface.OnClickListener()
    {
     @Override
     public void onClick(DialogInterface dialog, int which)
```

```
        {
            StringBuilder stringBuilder = new StringBuilder();
            for (int i = 0; i < arrayFruitSelected.length; i++)
            {
                if (arrayFruitSelected[i] == true)
               {
                    stringBuilder.append(arrayFruit[i] + "、");
               }
            }
Toast.makeText(MainActivity.this, stringBuilder.toString(), Toast.LENGTH_SHORT).show();
         }
      }).
setNegativeButton("取消", new DialogInterface.OnClickListener()
      {
          @Override
          public void onClick(DialogInterface dialog, int which)
          {
              //TODO: 自动存根法
        }
      }).
      create();
   alertDialog.show();
}
}
```

运行程序，效果如图 6-5 所示。



图 6-5　复选按钮对话框

6.2.6　自定义对话框

前面分别介绍了几种常用的 AlertDialog 对话框，下面将介绍怎样自定义对话框。

【例 6-5】 创建一个用户登录对话框。登录对话框中界面中包含三个输入框及三个按钮。其操作步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 AlertDialog_Custom。

(2) 打开 res\layout 目录下的 main.xml 布局文件，其代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc">
</RelativeLayout>
```

(3) 在 res\layout 目录下创建一个 login.xml 布局文件，在文件中声明两个 TextView 控件及两个 EditText 控件，其代码为：

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
```

```
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical" >
    <LinearLayout
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:gravity = "center" >
        <TextView
            android:layout_width = "0dip"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
            android:text = "用户名" />
        <EditText
            android:layout_width = "0dip"
            android:layout_height = "wrap_content"
            android:layout_weight = "1" />
    </LinearLayout>
    <LinearLayout
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:gravity = "center" >
        <TextView
            android:layout_width = "0dip"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
            android:text = "密码" />
        <EditText
            android:layout_width = "0dip"
            android:layout_height = "wrap_content"
            android:layout_weight = "1" />
    </LinearLayout>
</LinearLayout>
```

(4) 打开 src\fs. AlertDialog_Custom 包下的 MainActivity. java 文件，在文件中实现自定义对话框的创建，其代码为：

```
package fs.alertdialog_custom;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
public class MainActivity extends Activity {
/* 第一次调用活动 */
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  //取得自定义 View
  LayoutInflater layoutInflater = LayoutInflater.from(this);
  View myLoginView = layoutInflater.inflate(R.layout.login, null);
  Dialog alertDialog = new AlertDialog.Builder(this).
    setTitle("用户登录").
    setIcon(R.drawable.fb1).
```

```
        setView(myLoginView).
        setPositiveButton("登录", new DialogInterface.OnClickListener() {
         @Override
         public void onClick(DialogInterface dialog, int which) {
          //TODO 自动存根法
         }
        }).
        setNegativeButton("取消", new DialogInterface.OnClickListener() {
         @Override
         public void onClick(DialogInterface dialog, int which) {
          //TODO 自动存根法
         }
        }).
        create();
      alertDialog.show();
    }
    }
```

运行程序，效果如图 6-6 所示。



图 6-6　自定义对话框

6.3　进度条对话框

6.3.1　进度条对话框概述

当执行查询或加载某些比较耗时的数据时不想程序阻塞，一般使用开启线程的方式来做耗时的工作，而界面上或可供用户继续操作，或弹出一个窗口，提示用户当前程序执行的进度，这时就需要使用进度条对话框(ProgressDialog)。

ProgressDialog 分横向和圆形两种类型，通过设置样式改变，默认为圆形。

Activity 提供了一种方便管理的创建、保存、回复对话框的机制，使用这种机制将不会重复创建 Dialog，其中的方法为：

- onCreateDialog(int id)：创建一个对话框，只在第一次创建该 id 标识的 Dialog 时执行。
- onCreateDialog(int id,Bundle args)：同上，带参数。
- onPrepareDialog(int id,Dialog dialog)：在 onCreateDialog 之后，每次在对话框被显示前都执行。
- onPrepareDialog(int id,Dialog dialog,Bundle args)，同上，带参数。
- showDialog(int id)：显示对话框。
- showDialog(int id,Bundle args)：显示对话框，带参数。
- dismissDialog(int id)：隐藏对话框，不从 Activity 中移除，保留状态。
- removeDialog(int id)：隐藏对话框，并从 Activity 中移除，移除后再显示会重新调用 onCreateDialog 创建 Dialog 对象。

如果使用这些方法，Activity 将通过 getOwnerActivity()方法返回该 Activity 管理的对话框。

6.3.2 进度条对话框经典案例

【例 6-6】 利用 ProgressDialog 控件创建一个进度条对话框。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Progress_Dialog。

(2) 打开 res\layout 目录下的 main. xml 布局文件，在文件中声明一个 TextView 及两个 Button 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
< TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "@string/hello_world" />
< Button
    android:text = "圆形进度条"
    android:id = "@ + id/Button1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"/>
< Button
    android:text = "长型进度条"
    android:id = "@ + id/Button2"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"/>
</LinearLayout >
```

(3) 打开 src\fs. progress_dialog 包下的 MainActivity. java 文件，在文件创建一个圆形进度条对话框及一个长形进度条对话框，其代码为：

```
package fs.progress_dialog;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.DialogInterface;
```

```
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class MainActivity extends Activity {
    /* 第一次调用活动 */
    private ProgressDialog mpDialog;
    private Button btn1,btn2;
    private int mCount = 0;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btn1 = (Button) this.findViewById(R.id.Button1);
        btn2 = (Button) this.findViewById(R.id.Button2);
        btn1.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View v) {
                mpDialog = new ProgressDialog(MainActivity.this);
                //设置风格为圆形进度条
                mpDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
                mpDialog.setTitle("提示");                    //设置标题
                mpDialog.setIcon(R.drawable.fb1);             //设置图标
                mpDialog.setMessage("这是一个圆形进度条");
                mpDialog.setIndeterminate(false);             //设置进度条是否为不明确
                mpDialog.setCancelable(true);                //设置进度条是否可以按退回键取消
                mpDialog.setButton("确定", new DialogInterface.OnClickListener(){
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.cancel();
                    }
                });
                mpDialog.show();
            }
        });
        btn2.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View v) {
                mCount = 0;
                mpDialog = new ProgressDialog(MainActivity.this);
                mpDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
                mpDialog.setTitle("提示");
                mpDialog.setIcon(R.drawable.fb1);
                mpDialog.setMessage("这是一个长型进度条");
                mpDialog.setMax(100);
                mpDialog.setProgress(0);
                mpDialog.setSecondaryProgress(50);
                mpDialog.setIndeterminate(false);
                mpDialog.setCancelable(true);
                mpDialog.setButton("取消", new DialogInterface.OnClickListener(){
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.cancel();
                    }
                });
```

```
                new Thread(){
                    public void run(){
                        try{
                            while(mCount<=100){
                                mpDialog.setProgress(mCount++);
                                Thread.sleep(100);
                            }
                            mpDialog.cancel();
                        }catch(Exception ex){
                            mpDialog.cancel();
                        }
                    }
                }.start();
                mpDialog.show();
            }
        });
    }
}
```

运行程序，默认界面如图 6-7(a)所示；单击界面中的“圆形进度条”按钮时效果如图 6-7(b)所示；单击界面中的“长型进度条”按钮时效果如图 6-7(c)所示。



(a) 默认界面　　(b) 圆形进度条　　(c) 长型进度条

图 6-7　进度条对话

6.4　日期时间选择对话框

6.4.1　日期时间选择对话框概述

DatePickerDialog 类实现了日期选择对话框，这种对话框类似于日期选择控件，只不过是通过对话框的形式呈现给用户。Android 在 DatePickerDialog 类中进行了很好的封装，对于开发者来说，只需要简单地调用即可使用日期选择对话框。

日期选择对话框同样需要在 onCreateDialog 方法中进行设置，并通过 showDialog 方法来

呈现给用户。在此，主要使用 onDateSetListener 方法实现日期选择对话框。其语法格式为：

```
Public DatePickerDialog onDateSetListener()
```

TimePickerDialog 类实现了时间选择对话框，这种对话框类似于时间选择控件，只不过是通过对话框的形式呈现给用户的。Android 在 TimePickerDialog 类中进行了很好地封装，对于开发者来说，只需简单地调用即可使用时间选择对话框了。

时间选择对话框同样需要在 onCreateDialog 方法中进行设置，并通过 showDialog 方法来呈现给用户。此处主要使用 OnTimeSetListener 方法，其语法格式为：

```
Public TimerPickerDialog OnTimeSetListener()
```

6.4.2 日期时间选择对话框经典案例

【例 6-7】 设计一个含有日期选择对话框和时间选择对话框，其操作步骤如下：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 DateTime_Dialog。

(2) 打开 res\layout 目录下的 main.xml 文件，在布局文件中声明一个 AnalogClock 控件、一个 DigitalClock 控件、一个 EditText 控件及一个 Button 控件，其代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/LinearLayout01"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:background="#aabbcc">
<EditText
     android:id="@+id/et"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:editable="false"
     android:cursorVisible="false" />
  <Button
      android:text="日期对话框"
      android:id="@+id/dateBtn"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content" />
  <Button
      android:text="时间对话框"
      android:id="@+id/timeBtn"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content" />
  <DigitalClock
      android:text="@+id/digitalClock"
      android:textSize="20dip"
      android:gravity="center"
      android:id="@+id/DigitalClock1"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content" />
   <AnalogClock
       android:id="@+id/analogClock"
       android:gravity="center"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content" />
```

```
</LinearLayout>
```

（3）打开 src\fs.datetime_dialog 包下的 MainActivity.java 文件，在文件中实现日期选择对话框、时间选择对话及将对应的日期与时间显示在文本框中的功能，其代码为：

```
package fs.datetime_dialog;
import java.util.Calendar;
import android.app.Activity;
import android.app.DatePickerDialog;
import android.app.Dialog;
import android.app.TimePickerDialog;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.TimePicker;
public class MainActivity extends Activity {
    private Button dateBtn = null;
    private Button timeBtn = null;
    private EditText et = null;
    private final static int DATE_DIALOG = 0;
    private final static int TIME_DIALOG = 1;
    private Calendar c = null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        et = (EditText)findViewById(R.id.et);
        dateBtn = (Button) findViewById(R.id.dateBtn);
        timeBtn = (Button) findViewById(R.id.timeBtn);
        dateBtn.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v) {
                showDialog(DATE_DIALOG);
                }
            });
        timeBtn.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v) {
                showDialog(TIME_DIALOG);
                }
            });
        }
    /* 创建日期及时间选择对话框 */
    @Override
    protected Dialog onCreateDialog(int id) {
        Dialog dialog = null;
        switch (id) {
        case DATE_DIALOG:
            c = Calendar.getInstance();
    dialog = new DatePickerDialog(this, new DatePickerDialog.OnDateSetListener() {
        public void onDateSet(DatePicker dp, int year,int month, int dayOfMonth) {
    et.setText("您选择了: " + year + "年" + (month + 1) + "月" + dayOfMonth + "日");
                    }
                },
                c.get(Calendar.YEAR),
                //传入年份
```

```
                    c.get(Calendar.MONTH),
                    //传入月份
                    c.get(Calendar.DAY_OF_MONTH)
                    //传入天数
                    );
                break;
                case TIME_DIALOG:
                    c = Calendar.getInstance();
            dialog = new TimePickerDialog(this,new TimePickerDialog.OnTimeSetListener(){
                public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
                            et.setText("您选择了: " + hourOfDay + "时" + minute + "分");
                            }
                        },
                        c.get(Calendar.HOUR_OF_DAY),c.get(Calendar.MINUTE),false);
                    break;
                    }
            return dialog;
            }
    }
```

(4) 打开 AndroidManifest.xml 文件,设置权限,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.datetime_dialog"
    android:versionCode = "1"
    android:versionName = "1.0" >
    <uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    <application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        <activity
            android:name = "fs.datetime_dialog.MainActivity"
            android:label = "@string/app_name" >
            <intent - filter>
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter>
        </activity>
    </application>
    <!-- 设置权限 -->
    <uses - permission android:name = "android.permission.WRITE_CALENDAR" />
</manifest>
```

运行程序,默认界面如图 6-8(a)所示;当单击界面中的“日期对话框”按钮时即弹出日期选择对话框,如图 6-8(b)所示;选择对应的日期,单击对话框中的 Done 按钮时,即在文本框中显示对应的选择日期。当单击界面中的“时间对话框”按钮时即弹出时间选择对话框,效果如图 6-8(c);选择对应的时间,单击对话框中的 Done 按钮时,即在文本框中显示对应的选择时间,如图 6-8(d)所示。



(a) 默认界面　　(b) 日期选择对话框　　(c) 时间选择对话框　　(d) 显示选择的时间

图 6-8　日期时间选择对话框

6.5　风格窗口

6.5.1　风格窗口概述

PopupWindow 可以创建类似于对话框风格的窗口，不同于 AlertDialog 对话框，PopupWindow 弹出的位置可以很多变化，按照有无偏移分，可以分为无偏移和偏移两种；按照参照类型不同又可以分为相对某个控件(Anchor 锚)的位置和父容器内部的相对位置两种。其常用函数为：

- showAsDropDown(View anchor)：相对某个控件的位置(正左下方)，无偏移。
- showAsDropDown(View anchor, int xoff, int yoff)：相对某个控件的位置，有偏移(正数表示下方右边，负数表示(上方左边))。
- showAtLocation(View parent, int gravity, int x, int y)：父容器容器相对位置，如正中央 Gravity.CENTER、下方 Gravity.BOTTOM 等。

6.5.2　风格窗口经典案例

【例 6-8】 利用 PopupWindow 创建对话框风格的窗口，在界面中创建了 4 个按钮，用户单击这些按钮即分别在不同位置上显示 PopupWindow。其具体实现操作为：

(1) 在 Eclipse 环境下建立一个名为 Popup_W 的工程。

(2) 编写主布局文件，用于创建 4 个按钮。打开 res\Layout 目录下的 main.xml 文件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:Android = "http://schemas.android.com/apk/res/android"
    Android:id = "@ + id/layout"
    Android:orientation = "vertical"
    Android:layout_width = "fill_parent"
    Android:layout_height = "fill_parent">
< TextView
    Android:id = "@ + id/tv_showText"
```

```
        Android:layout_width = "fill_parent"
        Android:layout_height = "wrap_content"
        Android:gravity = "center"
        Android:text = "@string/hello_world"
        Android:textSize = "22px"/>
    <Button
        Android:id = "@ + id/bt_PopupWindow1"
        Android:text = "以自己为 Anchor,不偏移"
        Android:layout_width = "fill_parent"
        Android:layout_height = "wrap_content"/>
    <Button
        Android:id = "@ + id/bt_PopupWindow2"
        Android:text = "以自己为 Anchor,正下方"
        Android:layout_width = "fill_parent"
        Android:layout_height = "wrap_content"/>
    <Button
        Android:id = "@ + id/bt_PopupWindow3"
        Android:text = "以屏幕中心为参照,不偏移(正中间)"
        Android:layout_width = "fill_parent"
        Android:layout_height = "wrap_content" />
    <Button
        Android:id = "@ + id/bt_PopupWindow4"
        Android:text = "以屏幕下方为参照,下方中间"
        Android:layout_width = "fill_parent"
        Android:layout_height = "wrap_content"/>
    </LinearLayout>
```

(3) 编写 PopupWindow 布局文件,在 res\Layout 目录下创建一个名为 dialog.xml 的文件,其代码为:

```
    <?xml version = "1.0" encoding = "utf - 8"?>
    <LinearLayout xmlns:Android = "http://schemas.android.com/apk/res/android"
        Android:orientation = "vertical"
        Android:layout_width = "fill_parent"
        Android:layout_height = "fill_parent" >
    <TextView
        Android:id = "@ + id/tv_tip"
        Android:layout_width = "fill_parent"
        Android:layout_height = "wrap_content"
        Android:text = "请输入内容: "/>
    <EditText
        Android:id = "@ + id/et_text"
        Android:layout_width = "fill_parent"
        Android:layout_height = "wrap_content"/>
    <LinearLayout
        Android:gravity = "center_horizontal"
        Android:layout_width = "fill_parent"
        Android:layout_height = "fill_parent" >
    <Button
        Android:id = "@ + id/bt_ok"
        Android:text = "确定"
        Android:layout_width = "100px"
        Android:layout_height = "50px"/>
    <Button
        Android:id = "@ + id/bt_cancle"
        Android:text = "取消"
```

```
    Android:layout_width = "100px"
    Android:layout_height = "50px" />
</LinearLayout>
</LinearLayout>
```

(4) 编写 Activity 文件,打开 src/fs.popup_w 包下的 MainActivity.java 文件,其代码为:

```
package fs.popup_w;
import android.app.Activity;
import android.content.Context;
import android.content.SharedPreferences.Editor;
import android.os.Bundle;
import android.util.Log;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup.LayoutParams;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Gallery;
import android.widget.PopupWindow;
import android.widget.TextView;
public class MainActivity extends Activity
{
    //PopupWindow 属于不阻塞的对话框,AlertDialog 则是阻塞的
    private Button bt_popupWindow1;
    private Button bt_popupWindow2;
    private Button bt_popupWindow3;
    private Button bt_popupWindow4;
    private TextView tv_showText;
    private PopupWindow popupWindow;
    private int screenWidth;
    private int screenHeight;
    private int dialgoWidth;
    private int dialgoheight;
    /* 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        initView();
    }
    //初始化控件和响应事件
    private void initView()
    {
        bt_popupWindow1 = (Button)findViewById(R.id.bt_PopupWindow1);
        bt_popupWindow2 = (Button)findViewById(R.id.bt_PopupWindow2);
        bt_popupWindow3 = (Button)findViewById(R.id.bt_PopupWindow3);
        bt_popupWindow4 = (Button)findViewById(R.id.bt_PopupWindow4);
        tv_showText = (TextView)findViewById(R.id.tv_showText);
        bt_popupWindow1.setOnClickListener(new ClickEvent());
        bt_popupWindow2.setOnClickListener(new ClickEvent());
        bt_popupWindow3.setOnClickListener(new ClickEvent());
        bt_popupWindow4.setOnClickListener(new ClickEvent());
```

```
        }
        //按钮单击事件处理
        private class ClickEvent implements OnClickListener
        {
            @Override
            public void onClick(View v)
            {
                //TODO 自动存根法
                switch(v.getId())
                {
                case R.id.bt_PopupWindow1: //以自己为 Anchor,不偏移
                    getPopupWindow();
                    popupWindow.showAsDropDown(v);
                    break;
        //以自己为 Anchor,偏移(screenWidth - dialgoWidth)/2, 0)按钮正下方
                case R.id.bt_PopupWindow2:
                    getPopupWindow();
                    popupWindow.showAsDropDown(v, (screenWidth - dialgoWidth)/2, 0);
                    break;
                case R.id.bt_PopupWindow3: //以屏幕中心为参照,不偏移
                    getPopupWindow();
                    popupWindow.showAtLocation(findViewById(R.id.layout), Gravity.CENTER, 0, 0);
                    break;
    //以屏幕左下角为参照,偏移(screenWidth - dialgoWidth)/2, 0) 屏幕下方中央
                case R.id.bt_PopupWindow4:
                    getPopupWindow();
                    popupWindow.showAtLocation(findViewById(R.id.layout),
                            Gravity.BOTTOM, 0, 0);
                    break;
                default:
                    break;
                }
            }
        }
        //创建 PopupWindow
        protected void initPopuptWindow()
        {
            //TODO 自动存根法
            View popupWindow_view = getLayoutInflater().inflate(
            //获取自定义布局文件 dialog.xml 的视图
            R.layout.dialog, null,false);
            //创建 PopupWindow 实例
            popupWindow = new PopupWindow(popupWindow_view, 200, 150, true);
           //dialog.xml 视图里面的控件
            Button bt_ok = (Button)popupWindow_view.findViewById(R.id.bt_ok);
            Button bt_cancle = (Button)popupWindow_view.findViewById(R.id.bt_cancle);
            final EditText et_text = (EditText)popupWindow_view.findViewById(R.id.et_text);
            bt_ok.setOnClickListener(new OnClickListener()
            {
                @Override
                public void onClick(View v)
                {
                    //TODO 自动存根法
                    tv_showText.setText(et_text.getText());    //在标签里显示内容
                    popupWindow.dismiss();                     //对话框消失
```

```
            }
        });
        bt_cancle.setOnClickListener(new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                //TODO 自动存根法
                popupWindow.dismiss();
            }
        });
        //获取屏幕和对话框各自高宽
        screenWidth = MainActivity.this.getWindowManager().getDefaultDisplay().getWidth();
        screenHeight = MainActivity.this.getWindowManager().getDefaultDisplay().getHeight();
        dialgoWidth = popupWindow.getWidth();
        dialgoheight = popupWindow.getHeight();
    }
    //获取 PopupWindow 实例
    private void getPopupWindow()
    {
        if(null!= popupWindow)
        {
            popupWindow.dismiss();
            return;
        }else
        {
            initPopuptWindow();
        }
    }
    @Override
    protected void onPause()
    {
        //TODO: 自动存根法
        super.onPause();
        Log.e("ActivityState", "onPause");
    }
    @Override
    protected void onResume()
    {
        //TODO: 自动存根法
        super.onResume();
        Log.e("ActivityState", "onResume");
    }
}
```

运行程序,效果如图 6-9 所示。



图 6-9　PopupWindow 效果图

第 7 章 Android 菜单

要想让 Android 应用程序有更完善的用户体验,除了设计人性化的用户界面以外,添加一些菜单可以让应用程序在功能上更完备。Android 平台所提供的菜单大体上可分为两类:选项菜单(Options Menu)和上下文菜单(Context Menu)。

7.1 选 项 菜 单

7.1.1 选项菜单概述

当 Activity 在前台运行时,如果用户按下手机上的 Menu 键,此时就会在屏幕底端弹出相应的选项菜单。但这个功能是需要开发人员编程来实现的,如果在开发应用程序时没有实现该功能,那么程序运行时按下手机上的 Menu 键是不会起作用的。

对于携带图标的选项菜单,每次最多只能显示 6 个,当菜单选项多于 6 个时,将只显示前 5 个和 1 个扩展菜单选项,单击扩展菜单选项将弹出其余菜单项。扩展菜单项中将不会显示图标,但是可以显示单选框及复选框。

在 Android 中通过回调方法来创建菜单并处理菜单项按下的事件,这些回调方法及说明如表 7-1 所示。

表 7-1 选项菜单相关的回调方法及说明

方 法 名	描 述
onCreateOptionsMenu(Menu menu)	初始化选项菜单,该方法只在第一次显示菜单时调用,如果需要每次显示菜单时更新菜单项,则需要重写 onPrepareOptionsMenu(Menu)方法
public boolean onOptionsItemSelected(MenuItem item)	当选项菜单中某个选项被选中时调用该方法,默认是一个返回 false 的空实现
public void onOptionsMenu(Menu menu)	当选项菜单关闭时(或由于用户按下了返回键或选择了某个菜单项)调用该方法
public boolean onPrepareOptionsMenu(Menu menu)	为程序准备选项菜单,每次选项菜单显示前会调用该方法。可以通过该方法设置某些菜单项可用或不可用或修改菜单项的内容。重写该方法时需要返回 true,否则选项菜单将不会显示

提示: 除了使用开发回调方法 onOptionsItemSelected 来处理用户选中的菜单事件,还可以为每个菜单项 MenuItem 对象添加 OnMenuItemClickListener 监听器来处理菜单项选中事件。

开发选项菜单将主要用到 Menu、MenuItem 及 SubMenu,下面对它们进行简单介绍。

1. Menu 类

一个 Menu 对象代表一个菜单，在 Menu 对象中可以添加菜单项 MenuItem，也可以添加子菜单 SubMenu。在 Menu 中常用的方法如表 7-2 所示。

表 7-2　Menu 的常用方法及说明

方法名称	参数说明	方法说明
• MenuItem add(int groupId, int itemId, int order, CharSequence title); • MenuItem add(int groupId, int itemId, int order, int titleRes); • MenuItem add(CharSequence title); • MenuItem add(int titleRes)	groupId 为菜单项所在的组 id，通过分组可以对菜单项进行批量操作，如果菜单项不需要属于任何组，则传入 NONE； itemId 为唯一标识菜单项的 id，可传入 NONE； order 为菜单项的顺序，可以传入 NONE；title 为菜单项显示的文本内容； titleRes 为 String 对象的资源标识符	向 Menu 添加一个菜单项，返回 MenuItem 对象
• SubMenu addSubMenu(int titleRes); • SubMenu addSubMenu(int groupId, int itemId, int order, int titleRes); • SubMenu addSubMenu(CharSequence title); • SubMenu addSubMenu(int groupId, int itemId, int order, CharSequence title)	groupId 为子菜单所在的组 id，通过分组可以对子菜单进行批量操作，如果子菜单不需要属于任何组，则传入 NONE； itemId 为唯一标识子菜单的 id，可传入 NONE； order 为子菜单的顺序，可传入 NONE； title 为子菜单显示的文本内容； titleRes 为 String 对象的资源标识符	向 Menu 添加一个子菜单，返回 SubMenu 对象
void clear()	…	移除菜单中所有的子项
void close()	…	如果菜单正在显示，则关闭菜单
MenuItem findItem(int id)	id：MenuItem 的标识符	返回指定 id 的 MenuItem 对象
void removeGroup(int groupId)	groupId 为组 id	如果指定 id 的组不为空，则从菜单中移除该组
void removeItem(int id)	id 为 MenuItem 的 id	移除指定 id 的 MenuItem
int size()	…	返回 Menu 中菜单项的个数

2. MenuItem 对象

MenuItem 对象代表一个菜单项，通常 MenuItem 实例通过 Menu 的 add 方法获得，在 MenuItem 中常用的成员方法如表 7-3 所示。

3. SubMenu 对象

SubMenu 继承自 Menu，每个 SubMenu 实例代表一个子菜单，SubMenu 中常用的方法如表 7-4 所示。

表 7-3　选项菜单相关的回调方法及说明

方 法 名 称	参 数 说 明	方 法 说 明
setAlphabeticShortcut(char alphaChar)	alphaChar 为字母快捷键	设置 MenuItem 的字母快捷键
MenuItem setNumericShortcut (char numericChar)	numericChar 为数字快捷键	设置 MenuItem 的数字快捷键
MenuItem setIcon(Drawable icon)	icon 为图标 Drawable 对象	设置 MenuItem 的图标
MenuItem setIntent(Intent intent)	intent 为与 MenuItem 绑定的 Intent 对象	为 MenuItem 绑定 Intent 对象，当被选中时，将会调用 startActivity 方法处理相应的 Intent
setOnMenuItemClickListener(MenuItem. OnMenuItemClickListener menuItemClickListener)	menuItemClickListener 为监听器	为 MenuItem 设置自定义的监听器，一般情况下，使用回调方法 onOptionsItemSelected 会更有效
setShortcut (char numericChar, char alphaChar)	numericChar 为数字快捷键；alphaChar 为字母快捷键	为 MenuItem 设置数字快捷键和字母快捷键，当按下快捷键或按住 Alt 的同时按下快捷键时将会触发 MenuItem 的选中事件
setTitle(int title)	title 为标题的资源 id	为 MenuItem 设置标题
setTitle(CharSequence title)	title 为标题的名称	
setTitleCondensed (CharSequence title)	title 为 MenuItem 的缩略标题	设置 MenuItem 的缩略标题，当 MenuItem 不能显示全部的标题时，将显示缩略标题

表 7-4　SubMenu 中常用方法

方法名名称	参 数 说 明	方 法 说 明
setHeaderIcon(Drawable icon)	icon 为标题图标 Drawable 对象	设置子菜单的标题图标
setHeaderIcon(int iconRes)	iconRes 为标题图标的资源 id	
setHeaderTitle(int titleRes)	titleRes 为标题文本的资源 id	设置子菜单的标题
setHeaderTitle(CharSequence title)	title 为标题文本对象	
setIcon(Drawable icon)	icon 为图标 Drawable 对象	设置子菜单在父菜单中显示的图标
setIcon(int iconRes)	iconRes 为图标资源 id	
setHeaderView(View view)	view 为用于子菜单标题的 View 对象	设置指定的 View 对象为子菜单图标

7.1.2　选项菜单经典案例

下面通过案例来说明选项菜单及子菜单的用法。

【例 7-1】　实现接收用户在菜单中的选项并输出到文本框控件中的功能。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 MenuOptions。

(2) 打开 res\layout 目录下的 main.xml 布局文件，文件主要包括一个 ScrollView 控件，在控件中声明一个 TextView 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/LinearLayout01"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical" >
<!-- 声明一个线性布局 -->
    <ScrollView
        android:id = "@ + id/ScrollView01"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent" >
<!-- 声明 ScrollView 控件 -->
        <EditText
            android:id = "@ + id/EditText01"
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent"
            android:cursorVisible = "false"
            android:editable = "false"
            android:text = "@string/label" >
<!-- 声明一个 EditText 控件 -->
        </EditText>
    </ScrollView>
</LinearLayout>
```

(3) 打开 res\values 目录下的 strings.xml 文件，在文件中声明变量，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <string name = "app_name">MenuOptions</string>
    <string name = "action_settings">Settings</string>
    <string name = "hello_world">Hello world!</string>
    <string name = "label">您的选择为\n</string>
    <!-- 声明名为 label 的字符串资源 -->
    <string name = "gender">性别</string>
    <!-- 声明名为 gender 的字符串资源 -->
    <string name = "male">男</string>
    <!-- 声明名为 male 的字符串资源 -->
    <string name = "female">女</string>
    <!-- 声明名为 female 的字符串资源 -->
    <string name = "hobby">爱好</string>
    <!-- 声明名为 hobby 的字符串资源 -->
    <string name = "hobby1">游泳</string>
    <!-- 声明名为 hobby1 的字符串资源 -->
    <string name = "hobby2">唱歌</string>
    <!-- 声明名为 hobby2 的字符串资源 -->
    <string name = "hobby3">吃货</string>
    <!-- 声明名为 hobby3 的字符串资源 -->
    <string name = "ok">确定</string>
    <!-- 声明名为 ok 的字符串资源 -->
</resources>
```

(4) 打开 src\fs.limenuoptions 包下的 MainActivity.java 文件，用于实现菜单选项及菜单子选项，其代码为：

```
package fs.menuoptions;
import android.app.Activity;
import android.os.Bundle;
```

```
import android.view.Menu;
import android.view.MenuItem;
import android.view.MenuItem.OnMenuItemClickListener;
import android.view.SubMenu;
import android.widget.EditText;
//声明程序中菜单选项及子菜单的编号,编号必须是唯一的
public class MainActivity extends Activity {
    final int MENU_GENDER_MALE = 0;                         //性别为男选项编号
    final int MENU_GENDER_FEMALE = 1;                       //性别为女选项编号
    final int MENU_HOBBY1 = 2;                              //爱好 1 选项编号
    final int MENU_HOBBY2 = 3;                              //爱好 2 选项编号
    final int MENU_HOBBY3 = 4;                              //爱好 3 选项编号
    final int MENU_OK = 5;                                  //确定菜单选项编号
    final int MENU_GENDER = 6;                              //性别子菜单编号
    final int MENU_HOBBY = 7;                               //爱好子菜单编号每个菜单项目的编号
    //声明了程序中菜单选项组的编号,该编号也是唯一的
    final int GENDER_GROUP = 0;                             //性别子菜单项组的编号
    final int HOBBY_GROUP = 1;                              //爱好子菜单项组的编号
    final int MAIN_GROUP = 2;                               //外层总菜单项组的编号
    //声明并创建了 1 个 MenuItem 数组
    MenuItem[] miaHobby = new MenuItem[3];                  //爱好菜单项组
    MenuItem male = null;                                   //男性性别菜单项
    @Override
    //为重写的 onCreate 方法,该方法的主要功能是设置当前的屏幕
    public void onCreate(Bundle savedInstanceState) {       //重写 onCreate 方法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                      //设置当前屏幕
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu){
        //性别单选菜单项组菜单若编组就是单选菜单项组
        SubMenu subMenuGender = menu.addSubMenu(MAIN_GROUP,MENU_GENDER,0,R.string.gender);
        subMenuGender.setIcon(R.drawable.ic_launcher);
        subMenuGender.setHeaderIcon(R.drawable.ic_launcher);
        male = subMenuGender.add(GENDER_GROUP, MENU_GENDER_MALE, 0, R.string.male);
        male.setChecked(true);
        subMenuGender.add(GENDER_GROUP, MENU_GENDER_FEMALE, 0, R.string.female);
        //设置 GENDER_GROUP 组是可选择的,互斥的
        subMenuGender.setGroupCheckable(GENDER_GROUP, true,true);
        //爱好复选菜单项组
        SubMenu subMenuHobby = menu.addSubMenu(MAIN_GROUP,MENU_HOBBY,0,R.string.hobby);
        miaHobby[0] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY1, 0, R.string.hobby1);
        miaHobby[1] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY2, 0, R.string.hobby2);
        miaHobby[2] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY3, 0, R.string.hobby3);
        miaHobby[0].setCheckable(true);                     //设置菜单项为复选菜单项
        miaHobby[1].setCheckable(true);
        miaHobby[2].setCheckable(true);
        //确定菜单项
        MenuItem ok = menu.add(GENDER_GROUP + 2,MENU_OK,0,R.string.ok);
    //实现菜单项单击事件监听接口
    OnMenuItemClickListener lsn = new OnMenuItemClickListener(){
            public boolean onMenuItemClick(MenuItem item) {
                appendStateStr();
                return true;
            }
```

```
        };
        ok.setOnMenuItemClickListener(lsn);          //给确定菜单项添加监听器
        //给确定菜单项添加快捷键
        ok.setAlphabeticShortcut('o');               //设置字符快捷键
        //注意,同时设置多次时只有最后一个设置起作用
        return true;
    }
    @Override                                    //单选或复选菜单项选中状态变化后的回调方法
    public boolean onOptionsItemSelected(MenuItem mi){
        switch(mi.getItemId()){
          case MENU_GENDER_MALE:                   //单选菜单项状态的切换要自行写代码完成
           case MENU_GENDER_FEMALE:
                mi.setChecked(true);
                appendStateStr();                  //当有效项目变化时记录在文本区中
                break;
           case MENU_HOBBY1:                       //复选菜单项状态的切换要自行写代码完成
           case MENU_HOBBY2:
           case MENU_HOBBY3:
               mi.setChecked(!mi.isChecked());
               appendStateStr();                   //当有效项目变化时记录在文本区中
               break;
        }
        return true;
    }
    //获取当前选择状态的方法
    public void appendStateStr(){
        String result = "您选择的性别为: ";
        if(male.isChecked()){
            result = result + "男";
        }
        else{
            result = result + "女";
        }
        String hobbyStr = "";
        for(MenuItem mi:miaHobby){
            if(mi.isChecked()){
                hobbyStr = hobbyStr + mi.getTitle() + "、";
            }
        }
        if(hobbyStr.length()> 0){
    result = result + ",您的爱好为: " + hobbyStr.substring(0, hobbyStr.length() - 1) + ".\n";
        }
        else{
            result = result + ".\n";
        }
        EditText et = (EditText)MainActivity.this.findViewById(R.id.EditText1);
        et.append(result);
    }
}
```

运行程序,默认的界面如图 7-1(a)所示;当单击界面中的 Menu 键,弹出对应的菜单项,如图 7-1(b)所示。



(a) 默认界面　　(b) 菜单项

图 7-1　选项菜单使用

7.2　单选复选菜单项

7.2.1　单选复选菜单项概述

在某些时候,如果希望所创建的菜单项是单选菜单项或多选菜单,则可以调用 MenuItem 的如下方法。

- setCheckable(Boolean checkable):设置该菜单项是否可以被勾选。

调用上面的方法后的菜单项默认为多选菜单项。

如果希望将一组菜单里的菜单项都设为可勾选的菜单项,则可调用如下方法。

- setGroupCheckable(int group,Boolean checkable,Boolean exclusive):设置 group 组里的所有菜单项是否可勾选;如果将 exclusive 设为 true,那么它们将为一组单选菜单项。

除此之外,Android 还为 MenuItem 提供了如下方法来设置快捷键。

- setAlphabeticShortcut(char alphaChar):设置字母快捷键。
- setNumericShort(char numericChar):设置数字快捷键。
- setShortcut(char numericChar,char alphaChar):同时设置两种快捷键。

7.2.2　单选复选菜单项经典案例

【例 7-2】 演示单选菜单项或多选菜单。其具体操作步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 Check_Radio_Menu。

(2) 打开 res\layout 目录下的 main.xml 布局文件,在文件中声明三个 Button 控件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
```

```
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc" >
<Button
    android:id = "@+id/alertButton"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "菜单选择"/>
<Button
    android:id = "@+id/checkBoxButton"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "多选菜单选择" />
<Button
    android:id = "@+id/radioButton"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "单选菜单选择" />
</LinearLayout>
```

(3) 打开 src\fs.check_radio.menu 包下的 MainActivity.java 文件，在文件中实现菜单项、单选菜单项及复选菜单项。当选择相应的选项时，即弹出对应的 Toast 提示消息，其代码为：

```
package fs.check_radio_menu;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ListView;
import android.widget.Toast;
public class MainActivity extends Activity {
private String[] areas = new String[]{"全部","南国桃园", "祖庙地铁站", "建民路", "汉口路",
"汾江路", "碧桂园" };
private boolean[] areaState = new boolean[]{true, false, false, false, false, false,false};
private RadioOnClick radioOnClick = new RadioOnClick(1);
private ListView areaCheckListView;
private ListView areaRadioListView;
   private Button alertButton;
   private Button checkBoxButton;
   private Button radioButton;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        alertButton = (Button)findViewById(R.id.alertButton);
        checkBoxButton = (Button)findViewById(R.id.checkBoxButton);
        radioButton = (Button)findViewById(R.id.radioButton);
        alertButton.setOnClickListener(new AlertClickListener());
        checkBoxButton.setOnClickListener(new CheckBoxClickListener());
        radioButton.setOnClickListener(new RadioClickListener());
    }
    /*
```

```
         * 菜单弹出窗口
         */
      class AlertClickListener implements OnClickListener{
    @Override
    public void onClick(View v) {
     new AlertDialog.Builder(MainActivity.this).setTitle("选择区域").setItems(areas, new
DialogInterface.OnClickListener(){
        public void onClick(DialogInterface dialog, int which){
        Toast.makeText(MainActivity.this, "您已经选择了: " + which + ":" + areas[which],Toast.
LENGTH_LONG).show();
         dialog.dismiss();
        }
     }).show();
    }
      }
      /*
         * 多选框弹出菜单窗口
         */
      class CheckBoxClickListener implements OnClickListener{
    @Override
    public void onClick(View v) {
     AlertDialog ad = new AlertDialog.Builder(MainActivity.this)
     .setTitle("选择区域")
     .setMultiChoiceItems(areas,areaState,new DialogInterface.OnMultiChoiceClickListener(){
        public void onClick(DialogInterface dialog,int whichButton, boolean isChecked){
         //点击某个区域
        }
       }).setPositiveButton("确定",new DialogInterface.OnClickListener(){
        public void onClick(DialogInterface dialog,int whichButton){
         String s = "您选择了:";
         for (int i = 0; i < areas.length; i++){
          if (areaCheckListView.getCheckedItemPositions().get(i)){
           s += i + ":" + areaCheckListView.getAdapter().getItem(i) + " ";
          }else{
           areaCheckListView.getCheckedItemPositions().get(i,false);
          }
         }
         if (areaCheckListView.getCheckedItemPositions().size() > 0){
          Toast.makeText(MainActivity.this, s, Toast.LENGTH_LONG).show();
         }else{
           //没有选择
         }
         dialog.dismiss();
        }
       }).setNegativeButton("取消", null).create();
     areaCheckListView = ad.getListView();
     ad.show();
    }
      }
      /*
         * 单选弹出菜单窗口
         */
      class RadioClickListener implements OnClickListener {
```

```
@Override
public void onClick(View v) {
 AlertDialog ad = new AlertDialog.Builder(MainActivity.this).setTitle("选择区域")
 .setSingleChoiceItems(areas,radioOnClick.getIndex(),radioOnClick).create();
 areaRadioListView = ad.getListView();
 ad.show();
}
  }
  /*
   * 点击单选框事件
   */
  class RadioOnClick implements DialogInterface.OnClickListener{
private int index;

public RadioOnClick(int index){
 this.index = index;
}
public void setIndex(int index){
 this.index = index;
}
public int getIndex(){
 return index;
}
public void onClick(DialogInterface dialog, int whichButton){
  setIndex(whichButton);
  Toast.makeText(MainActivity.this, "您已经选择了: " + index + ":" + areas[index], Toast
.LENGTH_LONG).show();
  dialog.dismiss();
}
}
}
```

运行程序,得到如图 7-2(a)所示的默认界面;单击界面中的“菜单选择”按钮得到如图 7-2(b)所示的效果;单击界面中的“多选菜单选择”按钮得到如图 7-2(c)所示的效果;单击界面中的“单选菜单选择”按钮得到如图 7-2(d)所示的效果。



(a) 默认界面　(b) 菜单选择界面　(c) 多选菜单选择　(d) 单选菜单选择

图 7-2　单复选菜单项

7.3 上下文菜单

7.3.1 上下文菜单概述

上下文菜单(Context Menu)不同于选项菜单,该菜单不是 menu 键来触发,而是需要在某个内容上按下几秒后触发,以列表的方式显示菜单,可设置列表顶部的图标及标题,不支持条目图标和快捷键。需要显式地为某个内容注册,使用上下文菜单的主要方法如下。

- registerForContextMenu(View view):为某个内容注册菜单。
- onCreateContextMenu(ContextMenu menu,View v,ContextMenuInfo menuInfo):创建 ContextMenu,会在 menu 第一次显示时调用。
- onContextItemSelected(MenuItem item):菜单项被选中后处理选中的菜单项。
- onContextMenuClosed(Menu menu):菜单被关闭的事件。
- openContextMenu(View view):调用打开菜单。
- closeContextMenu():调用关闭菜单。

7.3.2 上下文菜单经典案例

【例 7-3】 实现上下文菜单。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 context_menu。

(2) 打开 res\layout 目录下的 main.xml 布局文件,在文件中声明一个 ListView 控件及一个 TextView 控件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = " # aabbcc" >
    < TextView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "Hello,Android"/>
    < ListView
        android:id = "@ + id/lv"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"/>
</LinearLayout >
```

(3) 打开 res\menu 目录下创建一个文件,用于实现变量定义,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< menu
  xmlns:android = "http://schemas.android.com/apk/res/android">
  < item android:id = "@ + id/add" android:title = "增加"/>
  < item android:id = "@ + id/update" android:title = "更新"/>
  < item android:id = "@ + id/delete" android:title = "删除"/>
</menu >
```

(4) 打开 src\fs.context_menu 包下的 MainActivity.java 文件,在文件中实现三个上下文

菜单,当选择某个选项时,即弹出对应的 Toast 提示消息,其代码为:

```
package fs.context_menu;
import java.util.ArrayList;
import java.util.List;
import android.app.Activity;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuInfo;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
public class MainActivity extends Activity {
    ListView lv;
    private ArrayAdapter<String> adapter;
    private List<String> alist = new ArrayList<String>();
    /* 第一次调用活动. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        lv = (ListView)findViewById(R.id.lv);
        alist.add("第一");
        alist.add("第二");
        alist.add("第三");
        adapter = new ArrayAdapter<String>(this,android.R.layout.simple_expandable_list_item
_1,alist);
        lv.setAdapter(adapter);
        //注册视图对象,即为 ListView 控件注册上下文菜单
        registerForContextMenu(lv);
    }
    /**
     * 创建上下文菜单选项
     */
    @Override
    public void onCreateContextMenu(ContextMenu menu, View v,
            ContextMenuInfo menuInfo) {
         /**
          * 1.通过手动添加来配置上下文菜单选项
          * menu.add(0, 1, 0, "修改");
          * menu.add(0, 2, 0, "删除");
          * 2.通过 xml 文件来配置上下文菜单选项
          */
        MenuInflater mInflater = getMenuInflater();
        mInflater.inflate(R.menu.cmenu, menu);
        super.onCreateContextMenu(menu, v, menuInfo);
    }
    /**
     * 当菜单某个选项被单击时调用该方法
     */
    @Override
    public boolean onContextItemSelected(MenuItem item) {
```

```
            switch(item.getItemId()){
            case 1:
            Toast.makeText(this, "你选择了手动修改", Toast.LENGTH_SHORT).show();
                break;
            case 2:
             Toast.makeText(this, "你选择了手动删除", Toast.LENGTH_SHORT).show();
                break;
            case R.id.add:
            Toast.makeText(this, "你选择了 XML 增加", Toast.LENGTH_SHORT).show();
                break;
            case R.id.update:
            Toast.makeText(this, "你选择了 XML 更新", Toast.LENGTH_SHORT).show();
                break;
            case R.id.delete:
            Toast.makeText(this, "你选择了 XML 删除", Toast.LENGTH_SHORT).show();
                break;
            }
            return super.onContextItemSelected(item);
        }
        /**
         * 当上下文菜单关闭时调用的方法
         */
        @Override
        public void onContextMenuClosed(Menu menu) {
            //TODO 自动存根法
            super.onContextMenuClosed(menu);
        }
    }
```

运行程序，默认界面如图 7-3(a)所示，当长按界面中的“第一”、“第二”个文本框时，将弹出对应的菜单，效果如图 7-3(b)和图 7-3(c)所示。



(a) 默认界面



(b) 第一个上下文菜单



(c) 第二个上下文菜单

图 7-3　上下文菜单项

7.4 子 菜 单

7.4.1 子菜单概述

在 Android 中除了使用菜单之外，还可以设置子菜单(SubMenu)。通过子菜单可以将相同类别的菜单命令归类，带来更好的用户体验。SubMenu 类提供了子菜单的一些操作。SubMenu 类承 Menu 类，每一个 SubMenu 对象代表了一个子菜单。

SubMenu 菜单的主要方法有：

- setIcon(int iconRes)：用于设置子菜单显示的图标。
- add(int titleRes)：向子菜单中添加菜单项。
- setOnMenuItemClickListener(MenuItem,OnMenuItemClickListener menuItemClick-Listener)：设置子菜单项的监听器。

7.4.2 子菜单经典案例

【例 7-4】 利用子菜单实现改变字体大小。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Sub_Menu。

(2) 编写布局文件，布局两个文本框控件。打开 res\layout 目录下的 main. xml 文件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:id = "@ + id/LL"
    android:background = "@drawable/bj">
    < TextView android:layout_height = "wrap_content"
    android:layout_width = "match_parent"
    android:id = "@ + id/textView1"
    android:text = "@string/tv1"/>
    < TextView android:text = "TextView"
    android:layout_height = "wrap_content"
    android:layout_width = "match_parent"
    android:id = "@ + id/textView2"/>
</LinearLayout >
```

(3) 编写 Activity 文件，实现子菜单。打开 src\com. example. sub_m 包下的 MainActivity . java 文件，其代码为：

```
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.SubMenu;
import android.widget.Toast;
public class MainActivity extends Activity
{
    /** 第一次调用活动. */
```

```
TextView tv = null;
@Override
public void onCreate(Bundle savedInstanceState)
{
    //onCreate 方法
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    tv = (TextView)findViewById(R.id.textView2);
    tv.setText("设置选项菜单!");                      //设置文本
}
@Override
public boolean onCreateOptionsMenu(Menu menu)
{
    //重载 onCreateOptionsMenu 方法
    //TODO 自动存根法
    MenuItem menu1 = menu.add(0,1,1,"黑色");          //菜单项 1
    MenuItem menu2 = menu.add(0,2,2,"红色");          //菜单项 2
    SubMenu menu3 = menu.addSubMenu(0,3,3,"字号");    //设置子菜单
    menu1.setIcon(R.drawable.icon);                   //设置图标
    menu2.setIcon(R.drawable.icon);                   //设置图标
    menu3.setIcon(R.drawable.icon);                   //设置子菜单
    MenuItem sub1 = menu3.add(1, 4, 4, "10 号");      //设置子菜单项 1
    MenuItem sub2 = menu3.add(1, 5, 5, "15 号");      //设置子菜单项 2
    MenuItem sub3 = menu3.add(1, 6, 6, "20 号");      //设置子菜单项 3
    MenuItem sub4 = menu3.add(1, 7, 7, "25 号");      //设置子菜单项 4
    sub1.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener()
    {
        //设置监听器
        @Override
        public boolean onMenuItemClick(MenuItem item)
        {
            //TODO 自动存根法
            tv.setTextSize(10);                       //设置字体大小
            return false;
        }
    });
    sub2.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener()
    {
        //设置监听器
        @Override
        public boolean onMenuItemClick(MenuItem item)
        {
            //TODO 自动存根法
            tv.setTextSize(15);                       //设置字体大小
            return false;
        }
    });
    sub3.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener()
    {
        //设置监听器
        @Override
        public boolean onMenuItemClick(MenuItem item)
        {
            //TODO 自动存根法
            tv.setTextSize(20);                       //设置字体大小
```

```
                return false;
            }
        });
        sub4.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener()
        {
            //设置监听器
            @Override
            public boolean onMenuItemClick(MenuItem item)
            {
                //TODO 自动存根法
                tv.setTextSize(25);                 //设置字体大小
                return false;
            }
        });
    return super.onCreateOptionsMenu(menu);
    }
}
```

运行程序,单击界面中的 Menu 按钮,在弹出的菜单中选择"字号",效果如图 7-4 所示。



图 7-4　子菜单项效果图

7.5　下拉菜单 Spinner

如果要动态增减 Spinner 下拉菜单选项,就必须利用 ArrayList(动态数组)的依赖性来完成。

本节使用 Spinner 的定义菜单实现了事件的交互处理,利用了 ArrayList(动态数组)的依赖性和动态增减 Spinner 下拉菜单中的选项。在具体实现上,将设计一个 EditText 输入框,当用户输入了文字并单击"添加"按钮时,就会将输入的值添加 Spinner 到下拉菜单的最后一项,接着 Spinner 会停留在刚添加的选项上,单击"删除"按钮则会删除选择的 Spinner 选项。

【例 7-5】 演示动态 Spinner 菜单。其具体实现操作步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 Spinner_menu。

(2) 打开 res\layout 目录下的 main.xml 布局文件,在文件中定义一个 EditText 控件、用

于定义需要添加或删除的菜单项、一个添加按钮、一个删除按钮和一个 Spinner 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
     android:background = "#aabbcc">
< EditText
    android:id = "@ + id/et"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"/>
< Button
    android:id = "@ + id/add"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "添加"/>
< Button
    android:id = "@ + id/remove"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "删除"/>
< Spinner
    android:id = "@ + id/sp"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"/>
</LinearLayout >
```

(3) 打开 res\values 目录下的 Strings. xml 文件，在 Strings. xml 中定义一个初始的数组，就是刚开始时 Spinner 显示的项目，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "app_name"> Spinner 动态菜单</string >
    < string name = "action_settings"> Settings </string >
    < string name = "hello_world"> Hello world! </string >
    < string - array name = "action">
        < item >上班</item >
        < item >睡觉</item >
        < item >上网</item >
        </string - array >
</resources >
```

(4) 打开 src\spinner_menu 包下的 MainActivity. java 文件，实现 Spinner 动态菜单，实现 Spinner 菜单的添加与删除，当在文本框中输入对应的菜单项时，单击"添加"按钮，即实现菜单项的添加，单击"删除"按钮即实现菜单项的删除，其代码为：

```
package fs.spinner_menu;
import java.util.ArrayList;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
```

```
import android.widget.EditText;
import android.widget.Spinner;
public class MainActivity extends Activity {
    /** 第一个调用活动 */
    EditText et;
    Button add,remove;
    Spinner sp;
    ArrayList<String> list = new ArrayList<String>();
    ArrayAdapter<String> adapter;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        et = (EditText)findViewById(R.id.et);
        add = (Button)findViewById(R.id.add);
        remove = (Button)findViewById(R.id.remove);
        sp = (Spinner)findViewById(R.id.sp);
        //获取相应对象
        String[] ls = getResources().getStringArray(R.array.action);
        //获取 XML 中定义的数组
        for(int i = 0;i<ls.length;i++){
            list.add(ls[i]);
            }
        //把数组导入到 ArrayList 中
        adapter = new ArrayAdapter<String>(this,android.R.layout.simple_spinner_item,list);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        //设置下拉菜单的风格
        sp.setAdapter(adapter);
        //绑定适配器
        sp.setPrompt("标题栏");
        //设置对话框标题栏
        add.setOnClickListener(new OnClickListener(){  //添加按钮监听器
            @Override
            public void onClick(View v) {
                //TODO 自动存根法
                adapter.add(et.getText().toString());
                //添加输入的项,add 后自动调用 notifyDataSetChanged()
                //如果需要指定位置,使用 insert(String s, int index)方法
                setTitle(String.valueOf(list.size()));
                //在标题输出添加后 list 的大小
                }
            });
remove.setOnClickListener(new OnClickListener(){        //删除按钮监听器
            @Override
            public void onClick(View v){
                //TODO 自动存根法
                adapter.remove(sp.getSelectedItem().toString());
                //删除当前选中项,删除后自动调用 notifyDataSetChanged()
                setTitle(String.valueOf(list.size()));
            }
        });
    }
}
```

运行程序,效果如图 7-5 所示。



图 7-5　Spinner 动态菜单

第 8 章 Android 程序组件

Android 应用程序使用 Java 语言来进行编写，在支持标准 Java 语言的同时，也根据自身结构设计了特有组件和机制。Android 中有 4 大组件，分别为 Activity、Service、BroadcastReceiver 和 Content Provider。组件和程序之间进行消息传递则使用 Intent。同时，针对线程之间的信息传递也提出了自己特有的通信机制。

8.1 活　　动

Android 应用程序由 4 大组件来构成最基本的框架。其中，活动(Activity)是最重要也是使用频率最高的组件。一个 Activity 通常是一个单独的全屏显示，其中有若干视图控件以及对应的事件处理。大部分应用程序都包含了多个 Activity，当多个 Activity 相互跳转切换时，就形成了一个 Activity 栈，以及 Activity 之间的数据交换。

8.1.1 活动概述

Activity 类继承 Context，提供和用户进行交互的可视化界面，类结构如图 8-1 所示。通过调用 setContentView()方法来展示需要显示的视图，调用 findViewById()方法和视图中的控件进行绑定。

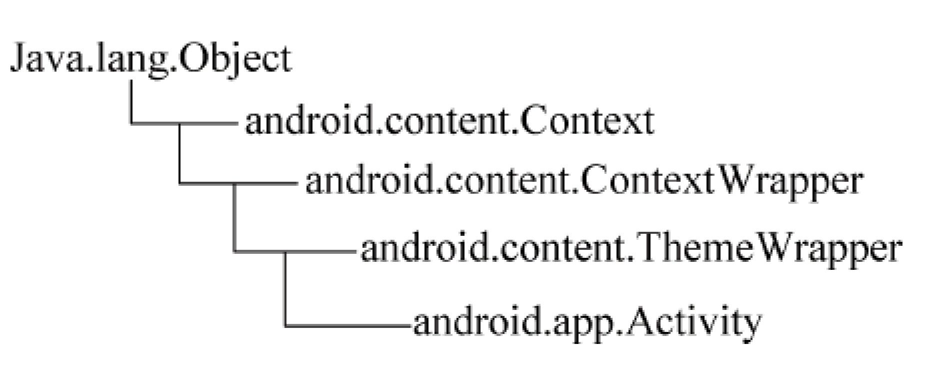


图 8-1　Activity 类结构

1. Activity 生命周期

程序也如同自然界的生物一样，有自己的生命周期。应用程序的生命周期即程序的存活时间。Android 为一构建在 Linux 之上的开源移动开发平台，在 Android 中，多数情况下每个程序都是在各自独立的 Linux 进程中运行的。当一个程序或其某些部分被请求时，它的进程就“出生”了，当这个程序没有必要再运行下去且系统需要回收这个进程的内存用于其他程序时，这个进程即为“死亡”了。可以看出，Android 程序的生命周期是由系统控制而非程序自身直接控制。这和编写桌面应用程序时的思维不同，一个桌面应用程序的进程也是在其他进程或用户请求时被创建，但是往往是在程序自身收到关闭请求后执行一个特定的动作(如从 main 函数中返回)而导致进程结束的。要想做好某种类型的程序或某种平台下程序的开发，最关键的就是要弄清楚这种类型的程序或整个平台下程序的一般工作模式并熟记在心。在 Android 中，程序的生命周期控制就是属于这个范畴。

开发者必须理解不同的应用程序组件，尤其是 Activity、Service 和 Intent Receiver。了解这些组件是如何影响应用程序的生命周期的，这非常重要。如果不能正确地使用这些组件，可能会导致系统终止正在执行重要任务的应用程序进程。

一个常见的进程生命周期漏洞的例子是 Intent Receiver(意图接收器),当 Intent Receiver 在 onReceive 方法中接收到一个 Intent(意图)时,其会启动一个线程,然后返回。一旦返回,系统将认为 Intent Receiver 不再处于活动状态,因而 Intent Receiver 所在的进程也就不再有用了(除非该进程中还有其他的组件处于活动状态)。因此,系统可能会在任意时刻终止该进程以回收占用的内存。这样进程中创建的那个线程也将被终止。解决这个问题的方法是从 Intent Receiver 中启动一个服务,让系统知道进程中还有处于活动状态的工作。为了使系统能够正确地决定在内存不足时应该终止进程,Android 根据每个进程中运行的组件及组件的状态把进程放入一个重要性分组(Importance Hierarchy)中。

一个 Android 程序的进程是何时被系统结束的呢?通俗地说,一个即将被系统关闭的程序是系统在内存不足(Low Memory)时,根据“重要性层次”选出来的“牺牲品”。一个进程的重要性是根据其中运行的部件和部件的状态决定的。各种进程按照重要性从高到低排列如图 8-2 所示。

图 8-2 Android 进程优先级效果图

1) 前台进程

前台进程是与用户正在交互的进程,也是 Android 系统中最重要的进程。处于前台进程一般包含以下 4 种情况:

- 进行中的 Activity 正在与用户进行交互;
- 进程服务被 Activity 调用,而且这个 Activity 正在与用户进行交互;
- 进程服务正在执行生命周期中的回调函数,如 onCreate()、onStart()或 onDestroy();
- 进程的 BroadcastReceiver 正在执行 onReceive()函数。

Android 系统为多任务操作系统,当系统中的多个前台进程同时运行时,如果出现资源不足的情况,此时 Android 内核将自动清除部分前台进程,保证最主要的用户界面能够及时响应操作。

2) 可见进程

可见进程是在屏幕上显示、但不在前台的程序。例如,一个前台进程以对话框的形式显示在该进程前面。这样的进程也很重要,它们只有在系统没有足够内存运行所有前台进程时,才会结束。

3) 服务进程

这样的进程在后台持续运行,如后台音乐播放、后台数据上传下载等。这样的进程对用户来说一般很有用,所以只有当系统没有足够内存来维持所有的前台和可见进程时,才会被结束。

4) 后台进程

这样的程序拥有一个用户不可见的 Activity。这样的程序在系统内存不足时,按照优先级的顺序结束。

5) 空进程

这样的进程不包含任何活动的程序部件。系统可能随时关闭这类进程。

从某种意义上讲,垃圾收集机制把程序员从“内存管理噩梦”中解放出来,而 Android 的进程生命周期管理机制把用户从“任务管理噩梦”中解放出来。Android 使用 Java 作为应用程序 API,并且结合其独特的生命周期管理机制同时为开发者和使用者提供最大程度的便利。

系统通过回调 Activity 中的方法来改变当前 Activity 的状态，这些方法一共有 7 个，分别为：

```
public class Activity extends ApplicationContent{
//创建时调用
protected void onCreate(Bundle savedInstanceState);
//启动时调用
protected void onStart();
//重新启动时调用
protected void onRestart();
//恢复时调用
protected void onResume;
//暂停时调用
protected void onPause();
//停止时调用
protected void onStop();
//销毁时调用
protected void onDestroy()
}
```

整个 Activity 的生命周期流程图如图 8-3 所示。

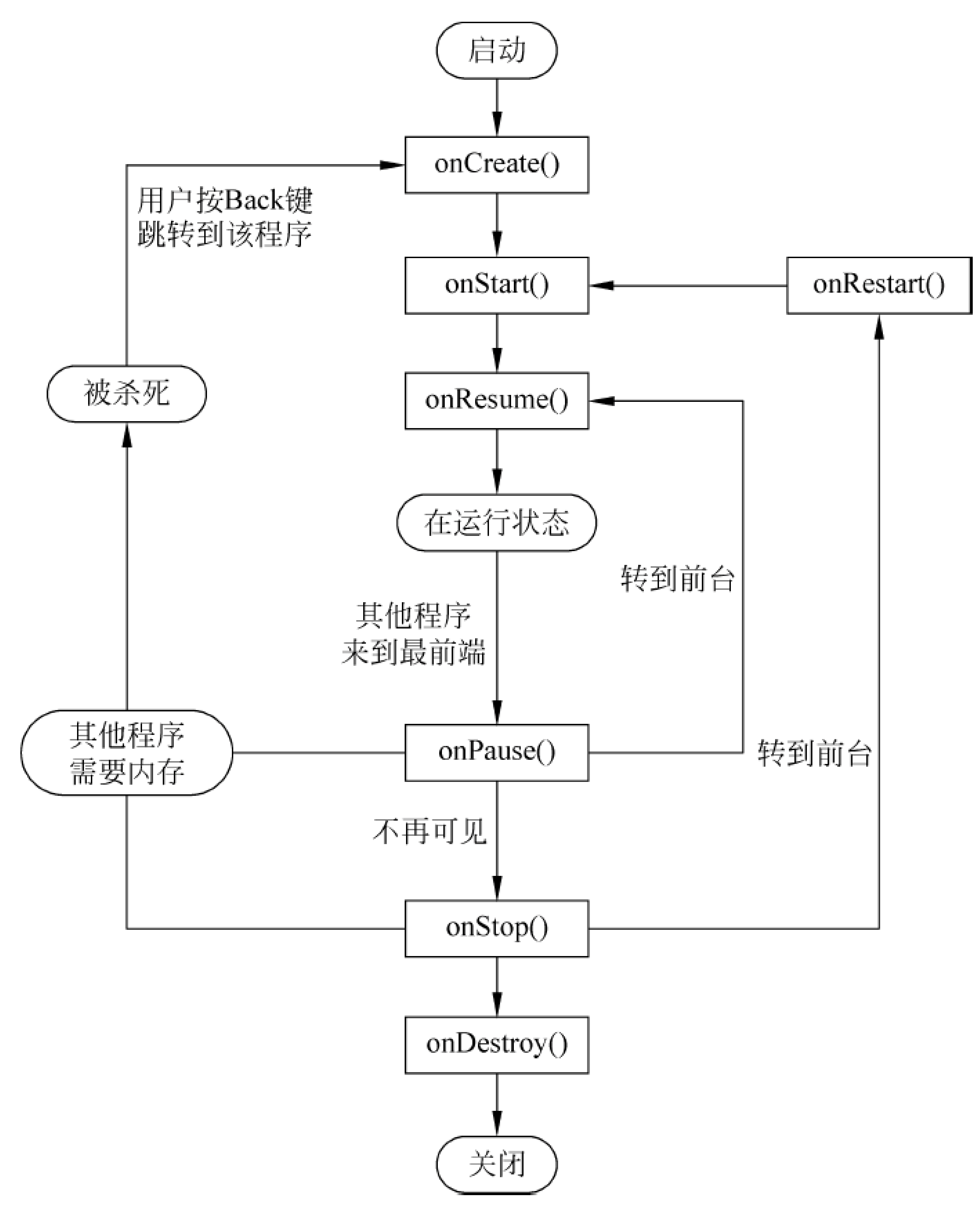


图 8-3　生命周期流程图

然后，通过重写这 7 种方法来观察一个 Activity 的生命周期变化。其具体操作步骤为：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 li8_1Activity_Cycle。

（2）打开 src\fs. li8_1activity_cycle 包下的 MainActivity. java 文件，在文件中实现 Activity 生命周期的 7 个方法，其代码为：

```
package fs.li8_1activity_cycle;
import android.app.Activity;
import android.content.res.Configuration;
import android.os.Bundle;
import android.util.Log;
import android.widget.Button;
import android.widget.EditText;
public class MainActivity extends Activity {
    String TAG = "Sample";
    /** 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);            //重写 onCreate()方法
        setContentView(R.layout.main);
        Log.i(TAG, "this is onCreate");
        EditText edt = (EditText)findViewById(R.id.edt);
    }
    @Override
    protected void onDestroy() {
        //TODO 自动存根法
        super.onDestroy();                             //重写 onDestroy()方法
        Log.i(TAG, "this is onDestroy");
    }
    @Override
    protected void onStart() {
        //TODO 自动存根法
        super.onStart();                               //重写 onStart()方法
        Log.i(TAG, "this is onStart");
    }
    @Override
    protected void onPause() {
        //TODO 自动存根法
        super.onPause();                               //onPause()方法
        Log.i(TAG, "this is onPause");
    }
    @Override
    protected void onRestart() {
        //TODO 自动存根法
        super.onRestart();                             //重写 onRestart()方法
        Log.i(TAG, "this is onRestart");
    }
    @Override
    protected void onResume() {
        //TODO 自动存根法
        super.onResume();                              //重写 onResume()方法
        Log.i(TAG, "this is onResume");
    }
    @Override
    protected void onStop() {
        //TODO 自动存根法
```

```
            super.onStop();                          //重写 onStop()方法
            Log.i(TAG, "this is onStop");
        }
        //改变横竖屏时
        @Override
        public void onConfigurationChanged(Configuration newConfig) {
            //TODO 自动存根法
            super.onConfigurationChanged(newConfig);
            Log.i(TAG, "this is onConfigurationChanged");
            if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
                //横屏时
                Log.i(TAG, "this is ORIENTATION_LANDSCAPE");
        } else if (newConfig.orientation == Configuration.ORIENTATION_PORTRAIT) {
                //竖屏时
                Log.i(TAG, "this is ORIENTATION_PORTRAIT");
            }
            //检测实体键盘的状态：推出或合上
            if (newConfig.hardKeyboardHidden == Configuration.HARDKEYBOARDHIDDEN_NO) {
                //实体键盘处于推出状态,在此处添加额外的处理代码
                Log.i(TAG, "this is HARDKEYBOARDHIDDEN_NO");
            } else if (newConfig.hardKeyboardHidden == Configuration.HARDKEYBOARDHIDDEN_YES) {
                //实体键盘处于合上状态,在此处添加额外的处理代码
                Log.i(TAG, "this is HARDKEYBOARDHIDDEN_YES");
            }
        }
    }
```

实现了这样的代码后,使用模拟器运行,默认此时是竖屏。在 LogCat 中输出如下信息：

```
06-01 11:18:24.108: I/Sample(1284): this is onCreate
06-01 11:18:24.108: I/Sample(1284): this is onStart
06-01 11:18:24.118: I/Sample(1284): this is onResume
06-01 11:18:24.638: W/IInputConnectionWrapper(1214): showStatusIcon on inactive InputConnection
06-01 11:18:24.698: D/gralloc_goldfish(1284): Emulator without GPU emulation detected.
06-01 11:18:24.848: I/ActivityManager(359): Displayed fs.li8_1activity_cycle/.MainActivity:
+3s237ms
06-01 11:18:25.538: I/Choreographer(1214): Skipped 136 frames! The application may be doing too
much work on its main thread.
```

2. 横、竖切换

由于 Activity 设备一般都带有重力感应系统,当改变设备的摆放位置时,系统会根据当前设备的位置来实现 Activity 的横、竖屏转换。但是,在横竖屏切换时,由于 Activity 的声明周期的原因,当前 Activity 会被销毁,然后重新创建一个新的 Activity,这样会导致输入的数据丢失。为了避免这样的情况,在实现横、竖屏切换时不销毁当前 Activity 的实现步骤为：

(1) 添加 Activity 属性。在 Mainfiest.xml 中的 Activity 声明中加入 android:configChanges="orientation|keyboardHidden"属性,这样就程序就可以在屏幕方向或键盘状态改变时做出相应处理,其代码为：

```
<activity
      android:configChanges = "orientation|keyboardHidden"
      android:name = "fs.li8_1activity_cycle.MainActivity"
      android:label = "@string/app_name" >
      <intent-filter>
          <action android:name = "android.intent.action.MAIN" />
```

```
            <category android:name = "android.intent.category.LAUNCHER" />
        </intent - filter>
    </activity>
```

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 TextView 控件及一个 EditText 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#aabbcc">
<TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "@string/hello_world" />
<EditText
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:id = "@ + id/edt"/>
</LinearLayout>
```

(3) 变化处理。添加了处理属性后，当屏幕方向改变或键盘状态改变时，系统会自己回调 Activity 中的函数进行处理，函数如下。

```
void onConfigurationChanged(Configuration newConfig)
```

其中，参数 newConfig 是改变后的状态信息。值得注意的是，在 onConfigurationChanged 中只会监测应用程序在 AndroidMainifest.xml 中通过 android:configChanges＝"xxxx"指定的配置类型的改动，对未指定的配置改变后，不会调用该函数进行处理而使用系统默认处理，即调用 onDestroy()销毁当前 Activity，然后重启一个新的 Activity 实例。在本实例中，先不对其做任何操作，只打印改变信息，其代码为：

```
//改变横竖屏时
@Override
public void onConfigurationChanged(Configuration newConfig) {
    //TODO 自动存根法
    super.onConfigurationChanged(newConfig);
    Log.i(TAG, "this is onConfigurationChanged");
    if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
        //横屏时
        Log.i(TAG, "this is ORIENTATION_LANDSCAPE");
} else if (newConfig.orientation == Configuration.ORIENTATION_PORTRAIT) {
        //竖屏时
        Log.i(TAG, "this is ORIENTATION_PORTRAIT");
    }
    //检测实体键盘的状态: 推出或者合上
if (newConfig.hardKeyboardHidden == Configuration.HARDKEYBOARDHIDDEN_NO) {
        //实体键盘处于推出状态,在此处添加额外的处理代码
        Log.i(TAG, "this is HARDKEYBOARDHIDDEN_NO");
    } else if (newConfig.hardKeyboardHidden == Configuration.HARDKEYBOARDHIDDEN_YES) {
        //实体键盘处于合上状态,在此处添加额外的处理代码
        Log.i(TAG, "this is HARDKEYBOARDHIDDEN_YES");
    }
}
```

调试的 LogCat 输出如下信息，可看出 Activity 创建运行后，没有被再次销毁。

```
06 - 01 11:21:50.408: I/ActivityManager(359): Displayed fs.li8_1activity_cycle/.MainActivity: +7s221ms
06 - 01 11:22:07.518: I/ActivityManager(359): Killing 1101:com.android.exchange/u0a24 (adj 15): empty for 1800s
06 - 01 11:22:07.528: I/ActivityManager(359): Killing 739:android.process.media/u0a4 (adj 15): empty for 1803s
06 - 01 11:30:08.228: D/ConnectivityService(359): Sampling interval elapsed, updating statistics ..
06 - 01 11:30:08.258: D/ConnectivityService(359): Done.
06 - 01 11:30:08.258: D/ConnectivityService(359): Setting timer for 720seconds
```

8.1.2 活动跳转

在应用程序中，一般不会只有一个 Activity，在多个 Activity 之间也进行跳转时大部分也会携带相关数据。下面通过一个调用系统电话拨号界面来讲解不同情况下 Activity 间跳转的处理。

在实例中，实现了拨打电话和发送邮件两个功能，在主界面中给出功能选择按钮。当选择不同的功能时，跳转到对应的详细界面。例如，选择“拨打电话”功能，则跳转到拨打电话的新界面中。在新界面中有需要输入被拨打电话的号码并且有跳转到系统拨号界面进行电话拨打和返回主界面两个功能选项。这两个功能选择分别使用带数据的跳转和有数据返回的跳转两种方式。

1. 不带数据跳转

不带数据的 Activity 跳转，即是从 ActivityA 直接跳转到 ActivityB，在 A 跳转到 B 时没有任何数据，同时从 B 返回 A 时也没有任何数据。接着，通过单击功能选择界面的“拨打电话”按钮直接跳转到拨打电话界面。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li8_2Activity_Jump。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中实现两个按钮布局，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<!-- 声明一个线性布局 -->
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#aabbcc">
<!-- 声明一个 TextView 控件 -->
    <TextView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "@string/hello_world" />
<!-- 声明一个 Button 控件 -->
    <Button
        android:id = "@ + id/call"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "拨打电话" />
<!-- 声明一个 Button 控件 -->
    <Button
        android:id = "@ + id/email"
        android:layout_width = "wrap_content"
```

```
        android:layout_height = "wrap_content"
        android:text = "发送邮件" />
</LinearLayout >
```

(3) 在 res\layout 目录中创建一个名为 call_phone. xml 文件。在该文件中定义一个垂直线性布局,声明一个 TextView 控件、一个 EditText 控件及两个 Button 控件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<!-- 声明一个线性布局 -->
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical" >
<!-- 声明一个 TextView 控件 -->
    < TextView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:background = "#FFFFFF"
        android:text = "请输入电话号码:" />
<!-- 声明一个 EditText 控件 -->
    < EditText
        android:id = "@ + id/edt_call_num"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content" />
<!-- 声明一个 Button 控件 -->
    < Button
        android:id = "@ + id/sys_call"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "拨打电话" />
<!-- 声明一个 Button 控件 -->
    < Button android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:id = "@ + id/Previous"
        android:text = "上一步"/>
</LinearLayout >
```

(4) 在 src\fs. li8_2activity_jump 包下创建一个继承 Activity 类的类,命名为 InputActivity . java,在文件中实现布局以及按键的处理等,与 MainActivity. java 相似用于实现对输入号码拨打电话的处理,其代码为:

```
public class InputActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //TODO 自动存根法
        super.onCreate(savedInstanceState);
        //设置布局
        setContentView(R.layout.call_phone);
}
```

(5) 注册声明。打开 AndroidManifest. xml 文件,在文件中注册声明新建的 Activity 类,其代码为:

```
  ⋮
        </activity >
        <!-- 注册声明新类 -->
        < activity android:name = ".InputActivity"/>
    </application >
</manifest >
```

(6) 实现跳转。打开 src\fs.li8_2activity_jump 包下的 MainActivity.java 文件，在文件输入需拨打的电话号码，然后通过单击拨打电话按钮来实现跳转到系统拨号界面，其代码为：

```
package fs.li8_2activity_jump;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
public class MainActivity extends Activity {
    Button btn_call,btn_email;
    static final int CALL_REQUEST = 0;
    /** 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btn_call = (Button)findViewById(R.id.call);
        btn_email = (Button)findViewById(R.id.email);
        btn_call.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                //TODO 自动存根法
                Intent call_intent = new Intent(MainActivity.this,InputActivity.class);
                //有返回结果的跳转
                startActivityForResult(call_intent, CALL_REQUEST);
            }
        });
    }
```

运行程序，出现如图 8-4(a)所示的功能选择界面，单击界面中的“拨打电话”按钮后，跳转到拨打电话的新界面，如图 8-4(b)所示。



(a) 功能选择界面　　(b) 拨打电话界面

图 8-4　Activity 不带数据跳转

2. 带数据的跳转

前面已经实现了一个界面到另一个界面的直接跳转，接着实现有数据的跳转。有数据的跳转，即从活动 A 跳转到活动 B 时应携带数据，但是从 B 返回 A 时却没有任何数据。实现从自定义的界面跳转到系统拨号界面，其实现步骤为：

修改 src\fs. li8_2activity_jump 包下的 InputActivity. java 文件，在文件中实现拨号跳转，其代码为：

```
        @Override
        public void onClick(View v) {
            //TODO 自动存根法
            String phoneString = edt_num.getText().toString();
            Pattern pattern = Pattern.compile("[0-9]*");
            if (pattern.matcher(phoneString).matches()) {
                //调用系统拨打电话
                Uri uri = Uri.parse("tel:" + phoneString);
                Intent sys_call_Intent = new Intent(Intent.ACTION_DIAL, uri);
                startActivity(sys_call_Intent);
            }
        }
    });
```

3. 有数据返回的跳转

有前面内容中，已经实现了无数据的跳转、有数据的跳转。接着，实现有数据返回的跳转。所谓有数据返回的跳转，即从活动 A 跳转到活动 B 时可以携带或不携带数据，但是从活动 B 返回活动 A 时必须携带数据。

在此，已经实现了从输入号码界面到系统拨号的返回跳转时，携带在输入框中输入的号码。其具体实现步骤为：

(1) 启动跳转。在无数据返回时，使用 startActivity()方法直接启动跳转。在有数据返回时，则使用另一个跳转方法 startActivityForResult(Intent intent，int requestCode)。其中，intent 为跳转的操作描述，requestCode 为自定义的请求标识。修改 src\fs. li8_2activity_jump 包下的 MainActivity. java 文件，实现跳转，其代码为：

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btn_call = (Button)findViewById(R.id.call);
        btn_email = (Button)findViewById(R.id.email);
        btn_call.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                //TODO 自动存根法
                Intent call_intent = new Intent(MainActivity.this,InputActivity.class);
                //有返回结果的跳转
                startActivityForResult(call_intent, CALL_REQUEST);
            }
        });
    }
```

(2) 从 B 返回数据。从活动 B 返回活动 A 时，需要携带数据。修改 InputActivity. java，实现携带数据，其代码为：

```
        btn_previous.setOnClickListener(new OnClickListener() {
```

```
                @Override
                public void onClick(View v) {
                    //TODO 自动存根法
                    on_Previous();
                }
            });
        }
        //处理 back 键
        @Override
        public boolean onKeyDown(int keyCode, KeyEvent event) {
            //TODO 自动存根法
            if (keyCode == KeyEvent.KEYCODE_BACK) {
                on_Previous();
                return true;
            }else {
                return super.onKeyDown(keyCode, event);
            }
        }
        void on_Previous(){
            Bundle bundle = new Bundle();
            String phoneString = edt_num.getText().toString();
            bundle.putString("PHONE_NUM", phoneString);
            InputActivity.this.setResult(RESULT_CANCELED, InputActivity.this.getIntent()
    .putExtras(bundle));
            InputActivity.this.finish();
        }
    }
```

对于从 B 返回数据的代码中,需要重点理解以下三点。

① 在 Intent 中使用 Bundle 类型来对附加数据进行描述。

Bundle 类似于哈希表 HashMap 的类型,保存一个键值对。使用如下方法来获取和添加数据。

```
Object get(String key)
void putString(String key,String value)
```

其中,参数 key 为键名;参数 value 为键值。

在 Intent 中,对附加数据 Bundle 的获取和添加使用如下方法。

```
Bundle getExtras()
Intent putExtras(Bundle extras)
```

其中,getExtras 方法返回 Bundle 类型数据;参数 extras 为添加的 Bundle,返回为 Intent。

② 获得数据后,返回上一个 Activity,使用方法为:

```
void setResult(int resultCode)
void setResult(int resultCode,Intent data)
```

其中,参数 resultCode 是结果标识,常用系统定义的 RESULT_CANCELED 或 RESULT_OK;参数 data 为返回的数据。

③ 需要特别注意的是,在 setResult 后,要调用 finish()销毁当前的 Activity,否则无法返回到原来的 Activity,致使无法执行原来 Activity 的 onActivityResult 函数。

(3) 返回数据处理。修改 MainActivity.java 文件,其代码为:

```
@Override
```

```
//结果返回处理
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    //TODO 自动存根法
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == CALL_REQUEST) {
        if (resultCode == RESULT_CANCELED) {
            Bundle bundle = data.getExtras();
            String phone_num = bundle.getString("PHONE_NUM");
            Toast.makeText(this, "拨打的号码是: " + phone_num, 1000).show();}
        }
    }
}
```

当数据从活动 B 返回到活动 A 时,A 需要对返回数据进行处理。需要重写 Activity 中的方法为:

```
onActivityResult(int requestCode, int resultCode, Intent data)
```

其中,参数 requestCode 即跳转时函数 startActivityForResult()中的请求标识;参数 resultCode 就是返回时函数 setResult()中的结果标识;参数 data 为具体的返回数据结果。本实例中,请求标识为 CALL_REQUEST,返回结果标识为 RESULT_CANCELED,返回的数据只有拨打的电话号码。

8.2 服 务

Service(服务)是 Android 系统中提供的 4 大组件之一,虽然没有 Activity 使用的频率高,但是在应用程序中与 Activity 同等重要。它是运行在后台的一种服务程序,一般生命周期较长,不直接与用户进行交互,因此没有可视化的界面。在服务中,最典型的应用实例是音乐播放器。在音乐播放器中,会提供一个或多个 Activity 界面给用户操作,但是音乐不会因为 Activity 的切换而停止,这时就需要服务来保证实现这样的效果。

Service 中为能自己启动运行,需要通过 Activity 或其他的 Context 对象来调用。启动服务有两种方式,分别为 Context. startService()和 Context. bindService()。这两种方式在启动过程和生命周期方面是有区别的。

8.2.1 服务生命周期

Service 生命周期有如下两种使用方式。

(1) 方法 1:Service 可以被启动或允许被启动直到有人停止或它自己停止。在这种模式下,它通过 Context. startService()方法开始,通过 Context. stopService()方法停止。它可以通过 Service. stopSelf()方法或 Service. stopSelfResult()方法来停止自己。无论调用了多少次的启动服务方法,只要调用一次 stopService()方法便可以停止服务。

(2) 方法 2:Service 可以通过定义的接口来编程,客户端建立一个与 Service 的链接,并使用此链接与 Service 进行通话。通过 Context. bindService()方法来绑定服务,用 Context . unbindService()方法来关闭服务。多个客户端可以绑定同一个服务。如果 Service 还未被启动,可以 bindService()方法启动服务。

这两种模式是完全独立的。可以绑定一个已经通过 startService()方法启动的服务。例如,一个后台播放音乐服务可以通过 startService()和一个 intend 对象来播放音乐。用户在播

放过程中要执行一些操作(如获取歌曲的一些信息),此时 Activity 可以通过调用 bindServices()方法与 Service 建立连接。这种情况下,stopServices()方法实际上不会停止服务,直到最后一次绑定关闭。

像一个 Activity 那样,Service 服务也有生命周期,也提供了事件回调函数:

```
void onCreate()
void onStart(Intent intent)
void onDestroy()
```

通过实现这三个生命周期方法,可以监听 Service 两个嵌套循环的生命周期。

(1) Service 整个生命周期。

Service 的整个生命周期是在 onCreate()和 onDestroy()方法之间。和 Activity 一样,在 onCreate()方法里初始化,在 onDestroy()方法里释放资源。例如,一个背景音乐播放服务可以在 onCreate()方法里播放,在 onDestroy()方法里停止。

(2) Service 活动的生命周期。

Service 的活动生命周期是在 onStart()之后,这个方法会处理通过 startServices()方法传递来的 Intent 对象。音乐 Service 可以通过打开 intent 对象来找到要播放的音乐,然后开始后台播放。

Service 停止时没有相应的回调方法,即没有 onStop()方法。onCreate()方法和 onDestroy()方法是针对所有的 Services,无论它们是否启动。通过 Context. startService()和 Context. bindService()方法。然而,只有通过 startService()方法启动的 Service 才会被调用 onStart()方法。如果一个 Service 允许别人绑定,那么需要实现以下额外的方法:

```
IBinder onBind(Intent intent)
boolean onUnbind(Intent intent)
void onRebind(Intent intent)
```

onBind()回调方法会继续传递通过 bindService()传递的 intent 对象。onUnbind()会处理传递给 unbindService()的 intent 对象。如果 Service 允许绑定,onBind()会返回客户端与服务互相联系的通信频道。如果建立了一个新的客户端与服务的链接,onUnbind()方法可以请求调用 onRebind()方法。

下面通过用 Java 代码来完成 Android 生命周期,其完整代码如下。

```
import android.app.Activity;
import android.os.Bundle;
public class MyActivity extends Activity
{
    //在完整生存期开始的时候调用
    @Override
    public void onCreate(Bundle icicle)
    {
        super.onCreate(icicle);
        //初始化一个活动
    }
    //在 onCreate 方法完成之后调用,用来恢复 UI 状态
    @Override
    public void onRestoreInstanceState(Bundle savedInstanceState)
    {
        super.onRestoreInstanceState(savedInstanceState);
        //从 savedInstanceState 中恢复 UI 状态
```

```
        //这个 Bundle 也被传递给 onCreate
    }
    //在一个活动进程后续的可见生存期之前调用
    @Override
    public void onRestart()
    {
        super.onRestart();
        //当知道这个进程中的活动已经可见后,载入改变
    }
    //在可见生存期开始时调用
    @Override
    public void onStart()
    {
        super.onStart();
        //既然活动可见,就应用任何要求的 UI 改变
    }
    //在活动状态生存期开始时调用
    @Override
    public void onResume()
    {
        super.onResume();
        //恢复活动所需要的任何暂停的 UI 更新、线程或进程
        //但是当不活动时,就挂起它们
    }
    //在活动状态生存期结束时调用,用来保存 UI 状态改变
    @Override
    public void onSaveInstanceState(Bundle savedInstanceState)
    {
        //把 UI 状态改变保存到 savedInstanceState
        //如果进程被销毁或重启,那么这个 Bundle 将被传递给
        //onCreate super.onSaveInstanceState(savedInstanceState);
    }
    //在活动状态生存期结束时调用
    @Override
    public void onPause()
    {
        /* 当活动不是前台的活动状态的活动时,挂起不需要更新的 UI 更新、线程或 CPU 密集的进程 */
        super.onPause();
    }
    //在可见生存期结束时调用
    @Override
    public void onStop()
    {
        /* 当进程不可见时,挂起不需要的剩下的 UI 更新、线程或处理,
           保存所有的编辑或者状态改变,因为这个进程可能会被销毁(从而为其他进程释放资源) */
        super.onStop();
    }
    //在完整生存期结束时调用
    @Override
    public void onDestroy()
    {
        //清空所有的资源,包括结束线程、关闭数据库连接等
        super.onDestroy();
    }
}
```

8.2.2 服务创建

由于 Service 是不能自己启动的，所以需要手动添加代码。整个过程和新建一个 Activity 类似，具体实现操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li8_3Service_Create。

(2) 注册声明。打开 AndroidManifest.xml 文件，在文件中注册声明新建的 Service，其代码为：

```
⋮
        </activity>
        <!-- 注册声明新建的 Service -->
      <service android:name = ".LocalService"/>
    </application>
</manifest>
```

(3) 打开 res\values 目录下的 strings.xml 文件，用于修改字符串，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <string name = "app_name">创建 Service</string>
    <string name = "action_settings">Settings</string>
    <string name = "hello_world">Hello world!</string>
    <string name = "local_service_started">开始局部的 service</string>
    <string name = "local_service_stopped">停止局部的 service</string>
    <string name = "local_service_label">样本局部的 Service</string>
    <string name = "bind_service">捆绑 Service</string>
    <string name = "unbind_service">不捆绑 Service</string>
    <string name = "local_service_connected">连接局部的 service</string>
    <string name = "local_service_disconnected">不连接 service</string>
</resources>
```

(4) 打开 res\layout 目录下的 main.xml 布局文件，在布局文件中声明 4 个 Button 控件、一个 TextView 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#aabbcc">
    <TextView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "@string/hello_world"/>
    <Button
        android:id = "@ + id/start"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:enabled = "true"
        android:text = "开启服务" />
    <Button
        android:id = "@ + id/stop"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:enabled = "true"
```

```
            android:text = "停止服务"/>
        < Button
            android:id = "@ + id/binded"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "bind 开启服务" />
        </Button >
        < Button
            android:id = "@ + id/unbind"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "bind 停止服务" />
</LinearLayout >
```

(5) 在 src\fs. li8_3service_create 包下创建一个类继承 Service，命名为 Second_Service. java，在Service 中有一系列与其生命周期相关的方法，这些方法有如下几种。

- abstract IBinder onBind(Intent intent)：必须实现的方法，返回一个绑定的接口给 Service。
- void onCreate()：当 Serivce 第一次被创建时，调用该方法。
- int onStartCommand(Intent intent，int flags，int startId)：当通过 startService()方法启动 Service 时，调用该方法。
- void onDestroy()：当 Service 结束不再使用时，调用该方法。
- boolean onUnbind(Intent intent)：当通过 bindService()方法启动 Service 时，取消绑定，调用该方法。

Second_Service. java 文件的代码为：

```
package fs. li8_3service_create;
import android. app. Notification;
import android. app. NotificationManager;
import android. app. PendingIntent;
import android. app. Service;
import android. content. Intent;
import android. os. Binder;
import android. os. IBinder;
import android. util. Log;
import android. widget. Toast;
public class Second_Service extends Service {
    private String TAG = "LOCALSERVICE";
    private NotificationManager mNM;
    private final IBinder mBinder = new LocalBinder();
    public class LocalBinder extends Binder {
        Second_Service getService() {
            return Second_Service. this;
        }
    }
    @Override
    public IBinder onBind(Intent intent) {
        //TODO 自动存根法
        Log. i(TAG, "this is onbind");
        return mBinder;
    }
    @Override
```

```
    public void onCreate() {
        mNM = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        Log.i(TAG, "this is oncreate");
        showNotification();
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.i(TAG, "Received start id " + startId + ": " + intent);
        return START_STICKY;
    }
    @Override
    public void onDestroy() {
        mNM.cancel(R.string.local_service_started);
        Log.i(TAG, "this is ondestroy");
        Toast
                .makeText(this, R.string.local_service_stopped,
                        Toast.LENGTH_SHORT).show();
    }
    @Override
    public boolean onUnbind(Intent intent) {
        //TODO 自动存根法
        Log.i(TAG, "this is onUnbind");
        return super.onUnbind(intent);
    }
    //显示 Notification
    private void showNotification() {
        CharSequence text = "Local service has started";
        Notification notification = new Notification(R.drawable.ic_launcher,
                text, System.currentTimeMillis());
        PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
                new Intent(this, MainActivity.class), 0);
        notification.setLatestEventInfo(this, "Local Service", text,
                contentIntent);
        mNM.notify(R.string.local_service_started, notification);
    }
}
```

通过上述步骤,实现了一个服务,接着来实现它的两种不同的开启方式。

8.2.3 服务开始

Service 不能自己启动,所以建立一个 Activity 来控制 Service 的启动与停止。使用两个按钮来实现开启服务和停止服务。其具体实现操作步骤如下。

打开 src\fs.li8_3service_create 包下的 MainActivity.java 文件,该文件用于启动与停止 Service,其代码为:

```
package fs.li8_3service_create;
import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.util.Log;
```

```
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //开始
        Button button = (Button)findViewById(R.id.start);
        button.setOnClickListener(mStartListener);
        button = (Button)findViewById(R.id.stop);
        button.setOnClickListener(mStopListener);
    }
    private OnClickListener mStartListener = new OnClickListener() {
        public void onClick(View v) {
            startService(new Intent(MainActivity.this,
                     Second_Service.class));
        }
    };
    private OnClickListener mStopListener = new OnClickListener() {
        public void onClick(View v) {
            stopService(new Intent(MainActivity.this, Second_Service.class));
        }
    };
```

对上述启动服务和停止服务代码,重点理解如下两点。

(1) 在 Activity 中,使用 startService 方式启动服务直接调用方法:

```
startService(Intent service)
```

其中,参数 service 是从当前上下文 Context 跳转到需要开启服务的 Intent。本案例中,具体实现如下。

```
startService(new Intent(Ex_Second_Service.this,
Second_Service.class));
```

(2) 停止服务服务直接调用方法:

```
boolean stopService(Intent name)
```

其中,参数 name 是需要停止的服务说明 Intent。本案例中,具体实现如下:

```
stopService(new Intent(MainActivity.this, Second
_Service.class));
```

运行程序,效果如图 8-5 所示。



图 8-5　开启服务界面

8.2.4　服务绑定

在前面内容中已介绍了 startService 的方式来启动一个服务,接着讲解使用 bindService 方式启动的服务。同样地,添加两个按钮用于开启和停止服务。其具体操作步骤为:

打开 src\fs. li8_3service_create 包下的 MainActivity. java 文件，在文件中添加以下代码为：

```
protected void onCreate(Bundle savedInstanceState) {      //重写 onCreate()方法
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);                        //设置界面布局
     //绑定
    Button bind_button = (Button)findViewById(R.id.binded);     //绑定控件
    bind_button.setOnClickListener(mBindListener);        //设置监听
    bind_button = (Button)findViewById(R.id.unbind);
    bind_button.setOnClickListener(mUnbindListener);
private OnClickListener mBindListener = new OnClickListener() {     //实现按钮监听
    public void onClick(View v) {
        doBindService();                                  //绑定服务
    }
};
   private OnClickListener mUnbindListener = new OnClickListener() {     //实现按钮监听
    public void onClick(View v) {
        doUnbindService();                                //取消绑定
    }
};
private boolean mIsBound;                                 //定义变量
private LocalService mBoundService;                       //定义服务
//实例化 ServiceConnection
private ServiceConnection mConnection = new ServiceConnection() {
    public void onServiceConnected(ComponentName className, IBinder service) {
        mBoundService = ((LocalService.LocalBinder)service).getService();
        Toast.makeText(MainActivity.this, R.string.local_service_connected,
                Toast.LENGTH_SHORT).show();               //显示提示信息
    }
//实例化断开服务
     public void onServiceDisconnected(ComponentName className) {
        mBoundService = null;
        Toast.makeText(MainActivity.this, R.string.local_service_disconnected,
                Toast.LENGTH_SHORT).show();               //显示提示信息
    }
};
void doBindService() {
    bindService(new Intent(MainActivity.this,             //定义绑定服务方法
    //绑定服务
    Second_1Service.class), mConnection, Context.BIND_AUTO_CREATE);
    mIsBound = true;                                      //已绑定
}
void doUnbindService() {                                  //定义解除绑定服务方法
    if (mIsBound) {
        unbindService(mConnection);                       //解除绑定服务
        mIsBound = false;                                 //未绑定
    }
}
```

在实现的代码中，理解以下两点：

(1) Activity 绑定、停止服务。在 Activity 中，使用绑定方式启动服务直接调用方法为：

```
boolean bindService(Intent service, ServiceConnection conn, int flags)
```

其中，参数 service 为描述跳转到服务的 Intent；参数 conn 为用于监测服务的状态接口；参数

flags 为绑定的操作选项，一般使用系统定义的 0、BIND_AUTO_CREATE、BIND_DEBUG_UNBINDor BIND_NOT_FOREGROUND。在本实例中，具体代码为：

```
bindService(new Intent(MainActivity.this,                //定义绑定服务方法
//绑定服务
Second_1Service.class), mConnection, Context.BIND_AUTO_CREATE);
```

停止服务直接调用方法如下。

```
void unbindService(ServiceConnection conn)
```

其中，参数 conn 为绑定时的检测服务状态的接口 ServiceConnection 类。

(2) 实现 ServiceConntextion。ServiceConntextion 类是用于检测服务状态的接口，实例化该类，必须实现如下两个方法。

```
void onServiceConnected(ComponentName name,IBinder service)
void onServiceDisconnected(ComponentName name)
```

其中，当服务被调用时将调用第一个方法；当服务停止时将调用第二个方法。

8.3 广　　播

广播接收者(BroadcastReceiver)用于接收广播 Intent，广播 Intent 的发送是通过调用 Context.sendBroadcast()、Context.sendOrderedBroadcast()来实现的。通常一个广播 Intent 可以被订阅了此 Intent 的多个广播接收者所接收。

广播是一种广泛运用的在应用程序之间传输信息的机制。BroadcastReceiver 是对发送出来的广播进行过滤接收并响应的一类组件；来自普通应用程序，如一个应用程序通知其他应用程序某些数据已经下载完毕。BroadcastReceiver 自身并不实现图形用户界面，但是当它收到某个通知后，BroadcastReceiver 可以启动 Activity 作为响应，或通过 NotificationMananger 提醒用户，或启动 Service 等。

8.3.1 广播生命周期

一个广播接收者有一个回调方法 void onReceive(Context curContext, Intent broadcastMsg)。当一个广播消息到达接收者时，Android 调用它的 onReceive()方法并传递给它包含消息的 Intent 对象。广播接收者被认为仅当它执行这个方法时是活跃的；当 onReceive()返回后，它是不活跃的。

有一个活跃的广播接收者的进程是受保护的，不会被杀死。但是，当别的进程需要占用内存时，系统可以在任何时候杀死仅有不活跃组件的进程。

这带来一个问题，当一个广播消息的响应费时的，应在独立的线程中做这些事，使用户界面其他组件主线程运行。如果 onReceive()衍生线程然后返回，整个进程，包括新的线程，被判定为不活跃的(除非进程中的其他应用程序组件是活跃的)，处于被杀的危险中。解决这个问题的方法是用 onReceive()启动一个服务，使系统知道进程中有活跃的工作在做。

BroadcastReceive 为广播接收器，和事件处理机制类似，只不过事件的处理机制是程序组件级别的，广播处理机制是系统级别的。

BroadcastReceiver 用于接收并处理广播通知(Broadcast Announcements)。多数的广播是系统发起的，如地域变换、电量不足、来电来信等。程序也可以播放一个广播。程序可以有

任意数量的BroadcastReceivers来响应它认为重要的通知。BroadcastReceiver可以通过多种方式通知用户：启动Activity、使用NotificationManager、开启背景灯、振动设备、播放声音等，最典型的是在状态栏显示一个图标，这样用户就可以单击它打开通知了。

通常某个应用或系统本身在某些事件(电池电量不足、来电、来短信)来临时会广播一个Intent，可以利用注册一个BroadcastReceiver来监听到这些Intent并获取Intent中的数据。

8.3.2 自定义广播

广播机制分为两部分，一部分是被广播的Intent；另一部分是接收该Intent的广播接收器(BroadcastReceiver)。在自定义的广播中，需要分别实现这两部分。其具体实现步骤为：

(1) 在Eclipse中创建一个Android应用项目，命名为li8_4Broadcast_Receive。

(2) 打开res\layout目录下的main.xml布局文件，在文件中实现主界面布局，定义自定义广播、取消自定义广播、发送自定义广播以及注册系统广播、取消注册系统广播三个按钮控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<!-- 声明一个线性布局 -->
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#aabbcc">
<!-- 声明一个线性布局 -->
< LinearLayout
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:layout_weight = "1"
    android:orientation = "vertical" >
<!-- 声明一个 TextView 控件 -->
< TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "@string/hello_world" />
<!-- 声明 5 个 Button 控件 -->
< Button
    android:id = "@ + id/regist_self"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:enabled = "true"
    android:text = "注册自定义广播"/>
< Button
    android:id = "@ + id/unregist_self"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:enabled = "true"
    android:text = "取消自定义广播"/>
  </LinearLayout >
    < Button
        android:id = "@ + id/send_self"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_gravity = "bottom"
```

```
        android:text = "发送自定义广播" />
</LinearLayout>
```

(3) 发送广播。打开 src\li8_4broadcast_receive 包下的 MainActivity.java 文件，在文件中实现界面布局、控件绑定以及发送广播的代码，其代码为：

```
package fs.li8_4broadcast_receive;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
public class MainActivity extends Activity {
    Button btn_registself, btn_unregistself, btn_sendbroadcast;
    Button btn_registsys, btn_unregistsys;
    broadcast_1 selfBroadcast;
    static final String SELF_ACTION = "com.sample.ex_broadcast.Internal";
    static final String SMS_ACTION = "android.provider.Telephony.SMS_RECEIVED";
    static final String TAG = "BROADCAST";
    private Context context;
    /** 第一次调用活动. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        context = this;
        selfBroadcast = new broadcast_1();
        btn_registself = (Button) findViewById(R.id.regist_self);
        btn_unregistself = (Button) findViewById(R.id.unregist_self);
        btn_sendbroadcast = (Button) findViewById(R.id.send_self);
        //注册自定义广播
        btn_registself.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                //TODO 自动存根法
                registerReceiver(selfBroadcast, new IntentFilter(SELF_ACTION));
                Log.i(TAG, "注册自定义广播");
                Toast.makeText(context, "注册自定义广播", 1000).show();
            }
        });
        //取消自定义 广播
        btn_unregistself.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                //TODO 自动存根法
                unregisterReceiver(selfBroadcast);
                Log.i(TAG, "注销广播");
                Toast.makeText(context, "注销自定义广播", 1000)
                        .show();
            }
        });
```

```
            //发送自定义广播
            btn_sendbroadcast.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    //TODO 自动存根法
                    Intent intent = new Intent(SELF_ACTION);
                    sendBroadcast(intent);
                    Log.i(TAG, "发送广播");
                    Toast.makeText(context, "发送注销广播", 1000).show();
                }
            });
        }
    }
```

在广播接收器中，通过 Intent 中不同的动作来区别接收到的广播是否需要处理，所以构造 Intent 使用的构造函数如下。

```
Intent(String action)
```

除此之外，在 Intent 中也可以定义其他附带的数据，用于处理接收器中的广播。定义了需要广播的 Intent 后，将该 Intent 广播到系统中，使用 Context 的方法如下。

```
void sendBroadcast(Intent intent)
void sendBroadcast(Intent intent,String receiverPermission)
void sendOrderedBroadcast(Intent intent,String receiverPermission)
```

其中，参数 intent 是需要广播的 Intent；参数 receiverPermission 是广播接收器需要的权限。在第二种方法中，只有应用程序具有一定的权限，才能接收处理该广播，一般为系统标准广播使用。

前两种方法发送的广播是无序广播，所有的广播接收器以无序方式运行，是完全异步的，往往在同一时间接收。这样效率较高，但是意味着接收者不能终止广播数据的传播。

第三种方法发送的广播是有序广播，一次传递给一个广播接收器，当该接收器处理完成后才会传递给下一个接收器。由于每个接收器依次执行，因此它可以传播到一个接收器，也可以完全终止传播该广播，从而使其他接收器无法接收到该广播。接收器的运行顺序可由匹配的意图过滤器(Intent-filter)的 android:priority 属性控件。

(4) 广播接收器。在 src\fs.li8_4broadcast_receive 包下创建一个类继承 BroadcastReceiver 类，用于实现广播接收后的处理，命名为 broadcast_1.java，其代码为：

```
package fs.li8_4broadcast_receive;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.widget.Toast;
public class broadcast_1 extends BroadcastReceiver{
    @Override
    public void onReceive(Context context, Intent intent) {  //重写 onReceiver 方法
        //TODO 自动存根法
        Log.i("BROADCAST", "broadcast_1 onreceive " + intent.toString());          //打印输出
        Toast.makeText(context, "broadcast_1 onreceive " + intent.toString(), 1000).show();
    }
}
```

运行程序，默认效果如图 8-6(a)所示。当单击界面中的"注册自定义广播"按钮时，效果如

图 8-6(b)所示；单击界面中的“取消自定义广播”按钮时，效果如图 8-6(c)所示；单击界面中的“发送自定义广播”按钮，效果如图 8-6(d)所示。



(a) 默认界面　(b) 显示注册自定义广播　(c) 注销自定义广播　(d) 发送自定义广播

图 8-6　自定义广播

8.3.3　短信广播

Android 中，不同进程之间传递信息要用到广播，有两种方式来实现，即为静态注册方式及动态注册方式。

1. 静态注册方式

静态注册即是在 Manifest.xml 中注册广播，不需要手动注销广播(如果广播未注销，程序退出时可能会出错)。

具体实现在 Manifest 的应用中添加：

```
<receiver android:name=".mEvtReceiver">
    <intent-filter>
    <action android:name="android.intent.action.BOOT_COMPLETED" />
    </intent-filter>
</receiver>
```

上面两个 android:name 分别是广播名和广播的动作(这里的动作是表示系统启动完成)，如果要自己发送一个广播，在代码中添加：

```
Intent i = new Intent("android.intent.action.BOOT_COMPLETED");
sendBroadcast(i);
```

这样，广播就发出去了，然后是接收。接收可以新建一个类，继承至 BroadcastReceiver，也可以建一个 BroadcastReceiver 的实例，然后得写 onReceive 方法，实现如下：

```
protected BroadcastReceiver mEvtReceiver = new BroadcastReceiver()
{
    @Override
        public void onReceive(Context context, Intent intent)
        {
        String action = intent.getAction();
        if (action.equals("android.intent.action.BOOT_COMPLETED"))
        {
            ⋮                          //做一些事件
```

```
        }
    }
};
```

下面通过一个完整的实例来介绍 BroadcastReceiver 的静态注册实现方法。其实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Broadcast_R1。

(2) 编写布局文件，布局一个文本框及一个按钮控件。打开 res\Layout 目录下的 main.xml 文件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
android:orientation = "vertical"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent"
android:background = "@drawable/bj" >
< TextView
android:layout_width = "fill_parent"
android:layout_height = "wrap_content"
android:text = "@string/hello_world" />
< Button
android:id = "@ + id/sned"
android:layout_width = "wrap_content"
android:layout_height = "wrap_content"
android:text = "@string/send"/>
</LinearLayout >
```

(3) 编写主 Activity 文件。打开 src\fs.boradcast_r1 包下的 MainActivity.java 文件，其代码为：

```
package fs.broadcast_r1;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class MainActivity extends Activity
{
    //定义 action 常量
    protected static final String ACTION = "com.example.broadcast_r1.RECEIVER_ACTION";
    //定义 Button 对象
    private Button btnBroadcast;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btnBroadcast = (Button)findViewById(R.id.sned);
        //为按钮设置单击监听器
        btnBroadcast.setOnClickListener(new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
```

```
                    //实例化 Intent
                    Intent intent = new Intent();
                    //设置 Intent 的 action 属性
                    intent.setAction(ACTION);
                    //发出广播
                     sendBroadcast(intent);
                }
            });
        }
    }
```

(4) 在 fs.broadcast_r1 包中定义一个 MyReceiver 类，继承于 BroadcastReceiver，覆盖 onReceive()方法。其实现代码为：

```
package fs.broadcast_r1;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
public class MyReceiver extends BroadcastReceiver
{
    //定义日志标签
    private static final String TAG = "Test";
    @Override
    public void onReceive(Context context, Intent intent)
    {
        //输出日志信息
        Log.i(TAG, "MyReceiver onReceive--->");
    }
}
```

(5) 授予权限。打开 AndroidManifest.xml 文件，代码为：

```
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.broadcast_r1"
    android:versionCode = "1"
    android:versionName = "1.0" >
    <uses-sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "15" />
    <application
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        <activity
            android:name = ".MainActivity"
            android:label = "@string/title_activity_main" >
            <intent-filter>
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name = "MyReceiver"/>
    </application>
</manifest>
```

运行程序，效果如图 8-7 所示。

静态注册方式的特点：不管改应用程序是否处于活动状态，都会进行监听。例如，某个程序监听内存的使用情况，当在手机上安装后，不管应用程序处于什么状态，都会执行改监听方法中的内容。



图 8-7　静态注册广播效果

2. 动态注册广播

动态注册方式在 Activity 里面调用函数来注册，和静态的内容类似。一个形参是 receiver；另一个是 IntentFilter，其中里面是要接收的 action。动态注册方式特点：在代码中进行注册后，当应用程序关闭后，就不再进行监听。

以短信接收为例，实现如下：

```
IntentFilter filter = new IntentFilter();
filter.addAction("android.provider.Telephony.SMS_RECEIVED");
registerReceiver(mEvtReceiver, filter);
```

这时注册了一个 recevier，名为 mEvtReceiver，然后同样用上面的方法以重写 onReceiver。

最后在程序的 onDestroy 中注销广播，实现如下：

```
@Override
public void onDestroy()
{
    super.onDestroy();
    unregisterReceiver(mPlayerEvtReceiver);
}
```

下面通过一个完整的实例来介绍 BroadcastReceiver 的动态注册实现方法。其实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Broadcast_R2。

(2) 编写布局文件，布局三个按钮控件。打开 res\Layout 目录下的 main. xml 文件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/bj">
< Button
    android:id = "@ + id/btnBroadcast"
    android:layout_width = "match_parent"
    android:layout_height = "wrap_content"
    android:text = "发送广播"/>
< Button
    android:id = "@ + id/btnregisterReceiver"
    android:layout_width = "match_parent"
    android:layout_height = "wrap_content"
    android:text = "注册广播接收器" />
< Button
    android:id = "@ + id/btnunregisterReceiver"
    android:layout_width = "match_parent"
    android:layout_height = "wrap_content"
```

```
    android:text = "注销广播接听器"/>
</LinearLayout >
```

(3) 编写主 Activity 文件,实现动态注册广播。打开 src\fs.broadcast_r2 包下的 MainActivity.java 文件,其代码为:

```
package fs.broadcast_r2;
import android.app.Activity;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class MainActivity extends Activity
{
    //定义 Action 常量
    protected static final String ACTION = "com.example.broadcast_r2.RECEIVER_ACTION";
    private Button btnBroadcast;
    private Button registerReceiver;
    private Button unregisterReceiver;
    private MyReceiver receiver;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btnBroadcast = (Button)findViewById(R.id.btnBroadcast);
        //创建事件监听器
        btnBroadcast.setOnClickListener(new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                Intent intent = new Intent();
                intent.setAction(ACTION);
                sendBroadcast(intent);
            }
        });
        registerReceiver = (Button)findViewById(R.id.btnregisterReceiver);
        //创建事件监听器
        registerReceiver.setOnClickListener(new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                receiver = new MyReceiver();
                IntentFilter filter = new IntentFilter();
                filter.addAction(ACTION);
                //动态注册 BroadcastReceiver
                registerReceiver(receiver, filter);
            }
        });
        unregisterReceiver = (Button)findViewById(R.id.btnunregisterReceiver);
        //创建事件监听器
        unregisterReceiver.setOnClickListener(new OnClickListener()
        {
            @Override
```

```
                public void onClick(View v)
                {
                    //注销 BroadcastReceiver
                    unregisterReceiver(receiver);
                }
            });
        }
    }
```

(4) 在 fs. broadcast_r2 包中定义一个 MyReceiver 类，继承于 BroadcastReceiver，覆盖 onReceive()方法。其实现代码为：

```
package fs.broadcast_r2;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
public class MyReceiver extends BroadcastReceiver
{
    //定义日志标签
    private static final String TAG = "Test";
    @Override
    public void onReceive(Context context, Intent intent)
    {
        //输出日志信息
        Log.i(TAG, "MyReceiver onReceive--->");
    }
}
```

(5) 授予权限。打开 AndroidManifest. xml 文件，添加代码为：

```
</application>
<uses-permission android:name="android.permission.
RECEIVE_SMS"/>
</manifest>
```

运行程序，效果如图 8-8 所示。

当首先单击图 8-8 中的"发送广播"按钮时，因为程序没有注册 BraodcastReceiver，所以 LogCat 没有输出任何信息。

如果先单击图 8-8 中的"注册广播接收器"按钮，再单击图 8-9 中的"发送广播"按钮，这时程序会动态地注册 BraodcastReceiver，之后调用 onReceive()方法，LogCat 输出信息见图 8-9。

当单击图 8-8 中的"注销广播接收器"按钮时，程序会注销 BraodcastReceiver，再单击图 8-8 中的"发送广播"按钮，LogCat 没有输出任何信息。



图 8-8　动态注册广播

			TID	Application	Tag	Text
I	09-23 08:14:46.545	2074	2074	com.example.broad...	Test	MyReceiver onReceive--->

图 8-9　LogCat 输出信息

8.4　消　　息

Android 的消息传递机制是另一种形式的“事件处理”，这种机制主要是为了解决 Android 应用的多线程问题——Android 平台不允许 Activity 新启动的线程访问该 Activity 里的界面组件，这样会导致新启动的线程无法动态改变界面组件的属性值。但在实际 Android 应用开发中，尤其是涉及动画的游戏开发中，需要让新启动的线程周期性地改变界面组件的属性值，这就需要借助于 Handler 的消息传递机制来实现。

每一个应用程序，都是一个单独的进程，运行于单独的 Dalvik 虚拟机实例中，再运行于单独的 Linux 进程中。每一个进程默认只有一个线程即 UI 主线程，因为它是以 UI 界面更新为主要任务的主线程，所以得名。

同样继承 Context 的 Activity 和 Service 都是运行在同一个线程里的，即 UI 主线程。这样它们之间是相互阻塞的，当 Service 运行较为费时的工作而 Activity 上的 UI 界面需要更新造成程序卡住时，就会导致程序被系统 Kill 掉。解决办法是 Service 需要打开更多的线程来运行费时的工作，如播放音乐、网络下载等。

打开的线程怎样与 UI 主线程互相协调工作呢，这就得需要用到 Handler 了。它最主要的作用是管理线程间的消息通信。

8.4.1　Handler 类概述

Android 系统通过 Looper、Handler 来实现消息循环机制。其消息循环是针对线程实现的，即每个线程都可以有自己的消息队列和消息循环。其中，Looper 负责管理线程的消息队列和消息循环；Handler 的作用是把消息加入特定的(Looper)消息队列中，并分发和处理该队列中的消息。

Handler 类位于 android.os 包，主要功能是完成 Activity 的 Widget 与应用程序中线程之间的交互。该类中常用的方法如表 8-1 所示。

表 8-1　Handler 类的常用方法

方法名称	说　　明
handleMessage(Message msg)	子类对象通过该方法接收信息
sendEmptyMessage(int what)	发送一个只含有 what 值的消息
sendMessage(int what)	发送消息到 Handler，通过 handleMessage 方法接收
post(Runnable r)	将一个线程添加到消息队列

开发带有 Handler 类的程序步骤如下。

(1) 在 Activity 或 Activity 的 Widget 中开发 Handler 类的对象，并重写 handleMessage 方法。

(2) 在新启的线程中调用 sendEmptyMessage 或 sendMessage 方法向 Handler 发送消息。

(3) Handler 类的对象用 handleMessage 方法接收消息，然后根据消息的不同执行不同的操作。

8.4.2 Handler 类经典案例

下面通过更新进度条及创建多彩的闪烁灯来介绍 Handler 类的使用。

【例 8-1】 本案例实现进度条更新，每间隔 1s 进度条前进 5%；当进度条达到 100%时，每间隔 1s 进度条回退 5%。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li8_5Handler_P。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 TextView 控件及一个 ProgressBar 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#aabbcc">
    < TextView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "@string/hello_world" />
    < ProgressBar
        android:id = "@ + id/progress"
        style = "?android:attr/progressBarStyleHorizontal"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content" />
</LinearLayout >
```

(3) 打开 src\fs. li8_5hndler_p 包下的 MainActivity. java 文件，在文件中添加 Handler 实现进度条变化，其代码为：

```
package fs.li8_5handler_p;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.widget.ProgressBar;
public class MainActivity extends Activity {
    final String TAG = "HANDLER";
    ProgressBar bar;
    final int INC = 1;
    final int DEC = 2;
    boolean is_running = false;
    Handler handler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            //TODO 自动存根法
            super.handleMessage(msg);
            switch (msg.what) {
            case INC:
                bar.incrementProgressBy(5);
        Log.i(TAG, "Thread id " + Thread.currentThread().getId() + ",handler INC");
                break;
            case DEC:
                bar.incrementProgressBy( - 5);
            Log.i(TAG, "Thread id " + Thread.currentThread().getId() + ",handler DEC");
```

```
                break;
            default:
        Log.i(TAG, "Thread id " + Thread.currentThread().getId() + ",handler DEFAULT");
                break;
            }
        }
    };
    /** 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        bar = (ProgressBar) findViewById(R.id.progress);
        Log.i(TAG, "oncreate thread id " + Thread.currentThread().getId());
    }
    @Override
    protected void onStart() {
        //TODO 自动存根法
        super.onStart();
        bar.setProgress(0);
        //使用 Handler 方式
        Thread handlerBarThread = new Thread(new Runnable() {
            @Override
            public void run() {
                //TODO 自动存根法
                //进度条增长
                for (int i = 0; i < 20 && is_running; i++) {
                    try {
                        Thread.sleep(1000);
                        Message msg = new Message();
                        msg.what = INC;
                        handler.sendMessage(msg);
        Log.i(TAG,"Thread id " + Thread.currentThread().getId() + ",sendmessage INC");
                    } catch (Exception e) {
                        //TODO 异常处理
                    }
                }
                //进度条减少
                for (int i = 0; i < 20 && is_running; i++) {
                    try {
                        Thread.sleep(1000);
                        Message msg = new Message();
                        msg.what = DEC;
                        handler.sendMessage(msg);
                        Log.i(TAG,"Thread id " + Thread.currentThread().getId() + ",
sendmessage DEC");
                    } catch (Exception e) {
                        //TODO 异常处理
                    }
                }
            }
        });
        is_running = true;
        handlerBarThread.start();
    }
    @Override
```

```
    protected void onStop() {
        //TODO 自动存根法
        super.onStop();
        is_running = false;
    }
}
```

对于 Handler 的实现代码中，理解以下几点：

- 对于 Looper。Activity 是一个 UI 线程，运行于主线程中，Android 系统在启动时会为 Activity 创建一个消息队列和消息循环。所以，一般情况下不用创建消息循环。但是，创建非 UI 线程默认是没有消息队列和消息循环的，如果想让该线程具有消息队列和消息循环，需要在线程中首先调用 Looper.prepare()来创建消息队列，然后调用 Looper.loop()进入消息循环。
- Handler 的方法。Handler 负责分发和处理消息循环中的消息。实现一个 Handler 类，必须实现 Handler 中的消息处理函数。

```
void hanldeMessage(Message msg)
```

其中，参数 msg 是在消息队列中消息类 Message，包含描述和任意数据对象；使用 Message 类中的 what 变量来定义消息代码，以使收件人能识别此消息。

- 消息发送线程。需要新建一个线程用于完成耗时的操作并及时向消息循环队列中发送消息包。

运行程序，效果如图 8-10 所示。

输出的调度信息如下所示：

```
06-03 04:58:47.440: I/HANDLER(1295): oncreate thread id 1
06-03 04:58:47.940: D/gralloc_goldfish(1295): Emulator without GPU emulation detected.
06-03 04:58:48.070: I/ActivityManager(371): Displayed fs.li8_5handler_p/.MainActivity:
+2s739ms
06-03 04:58:48.470: I/HANDLER(1295): Thread id 96,sendmessage INC
06-03 04:58:48.480: I/HANDLER(1295): Thread id 1,handler INC
06-03 04:58:48.560: I/Choreographer(371): Skipped 50 frames! The application may be doing too
much work on its main thread.
⋮
```

由以上信息可以看出，Activity 在创建函数 onCreate()中的线程号为 1，发送消息的线程号为 96，而 Handler 处理线程号为 1。由于消息处理是在主线程中处理的，所以在消息处理函数中可以安全地调用主线程中的任何资源，包括刷新界面。工作线程和主线程运行在不同的线程中，所以必须要注意这两个线程间的竞争关系，发送消息和处理消息不一定会交错进行。在发送多个信息后，Handler 才逐个处理信息。

在 Android 中除了提供 Handler 的消息循环机制外，还提供了一些有别于线程的处理方法。下面通过创建闪烁灯来说明 Handler 类的其他使用方法。



图 8-10　进度条更新

【例 8-2】 利用 Android 创建多彩的闪烁灯。其具体的实现操作为：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 colorright。

(2) 打开 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，并为默认添加的线性布局管理器设置 id 属性，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/bj1"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
</LinearLayout >
```

(3) 在 res\values 目录下，创建一个颜色资源文件，命名为 color.xml，在该文件中，定义 7 个颜色资源，名称依次为 color1，color2，…，color7，颜色值分别为赤、橙、黄、绿、青、蓝、紫，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < color name = "color1"># ffff0000 </color >
    < color name = "color2"># ffff6600 </color >
    < color name = "color3"># ffffff00 </color >
    < color name = "color4"># ff00ff00 </color >
    < color name = "color5"># ff00ffff </color >
    < color name = "color6"># ff0000ff </color >
    < color name = "color7"># ff6600ff </color >
</resources >
```

(4) 打开默认创建的 MainActivity，其中对相应的颜色已作说明，文件的主要代码为：

```
public class MainActivity extends Activity
{   //声明程序中所需的成员变量
    private Handler handler;                          //创建 Handler 对象
    private static LinearLayout linearLayout;         //整体布局
    public static TextView[ ] tv = new TextView[14];  //TextView 数组
    int[ ] bgColor = new int[ ]{R.color.color1,R.color.color2,R.color.color3,
    R.color.color4,R.color.color5,R.color.color6,R.color.color7};    //使用颜色资源
    private int index = 0;                            //当前的颜色值
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        /* 首先获取线性布局管理器，然后获取屏幕的高度，接着再通过一个 for 循环创建 14 个文本
框组件，并添加线性布局管理器中 */
        linearLayout = (LinearLayout)findViewById(R.id.bj1);     //获取线性布局管理器
        int height = this.getResources().getDisplayMetrics().heightPixels;
       //获取屏幕的高度
       for(int i = 0;i < tv.length;i++){
            tv[i] = new TextView(this);                //创建一个文本框对象
            //设置文本框的宽度
             tv[i].setWidth(this.getResources().getDisplayMetrics().widthPixels); tv[i].
setHeight(height/tv.length);                          //设置文本框的高度
            linearLayout.addView(tv[i]);            //将 TextView 组件添加到线性布局管理器中
        }
        /* 创建一个 Handler 对象，在重写的 handleMessage()方法中，为每个文本框设置背景颜色，该
背景颜色从颜色数组中随机获取 */
```

```
        handler = new Handler() {
            @Override
            public void handleMessage(Message msg) {
                int temp = 0;                              //临时变量
                if (msg.what == 0x101) {
                    for(int i = 0;i < tv.length;i++){
                        //产生一个随机数
                        temp = new Random().nextInt(bgColor.length);
                                                           //去掉重复的并且相邻的颜色
                        if(index == temp){
                            temp++;
                            if(temp == bgColor.length){
                                temp = 0;
                            }
                        }
                        index = temp; tv[i].setBackgroundColor(getResources().getColor
(bgColor[index]));                                         //为文本框设置背景
                    }
                }
                super.handleMessage(msg);
            }
        };
        /* 创建并开启一个新线程,在重写的 run()方法中实现一个循环,在该循环中,先获取一个
Message 对象,并为其设置一个消息标识,然后发送消息,最后让线程休眠 1s */
        Thread t = new Thread(new Runnable()
        {
            @Override
            public void run() {
                while (!Thread.currentThread().isInterrupted()) {
                    Message m = handler.obtainMessage();      //获取一个 Message
                    m.what = 0x101;                           //设置消息标识
                    handler.sendMessage(m);                   //发送消息
                    try {
                        Thread.sleep(new Random().nextInt(1000));     //休眠 1 秒钟
                    } catch (InterruptedException e) {
                        e.printStackTrace();                  //输出异常信息
                    }
                }
            }
        });
        t.start();                                            //开启线程
    }
}
```

(5) 在 AndroidMainfest.xml 文件的<Activity>标记中,设置 android:theme 属性,实现全屏显示,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.colorright"
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    < application
```

```
                android:allowBackup = "true"
                android:icon = "@drawable/ic_launcher"
                android:label = "@string/app_name"
                android:theme = "@style/AppTheme" >
                <Activity
                    android:name = ".MainActivity"
                    android:label = "@string/app_name"
                    android:theme = "@android:style/Theme.Black.NoTitleBar">
                    <intent - filter >
                        <action android:name = "android.intent.action.MAIN" />
                        <category android:name = "android.intent.category.LAUNCHER" />
                    </intent - filter >
                </Activity>
            </application>
        </manifest>
```

运行程序，将全屏显示一个多彩的闪烁灯，即它可以不断地变换颜色，效果如图 8-11 所示。



图 8-11　多彩闪烁灯

8.5　意　　图

一个 Android 程序由多个组件组成，各个组件之间使用意图(Intent)进行通信。Intent 对象中包含组件名称、动作、数据等内容。根据 Intent 中的内容，Android 系统可以启动需要的组件。

Intent 有显式和隐式之分，显式的 Intent 是根据组件的名称直接启动要启动的组件，如 Service 或 Activity；隐式的 Intent 通过配置的 Action、Category、Data 来找到匹配的组件并启动。

8.5.1　意图属性

要在不同的 Activity 之间传递数据，就要在 Intent 中包含相应的内容。一般来说，在数据

中最基本的包括如下两点。

- Action：用来指明要实施的动作是什么，如 ACTION_VIEW、ACTION_EDIT 等。
- Data：要实施的具体数据，一般由一个 URI 变量来表示。

例如：

```
ACTION_VIEW content:                          //显示 identifier 为 1 的联系人的信息
ACTION_CALL Content:⋮
ACTION_VIEW content:                          //给该联系人拨打电话
```

下面将详细介绍 Intent 的各属性值，以及 Android 根据不同属性值来启动相应的组件。

1. Component 属性

Intent 的 Component 属性需要接受一个 ComponentName 对象，ComponentName 语法格式为：

- ComponentName(String pkg,String cls)：创建 pkg 所在包下的 cls 所对应的组件。
- ComponentName(Context pkg,String cls)：创建 pkg 所对应的包下的 cls 类所对应的组件。
- ComponentName(Context pkg,Class<?>cls)：创建 pkg 所对应的包下的 cls 类所对应的组件。

上面的语法格式本质即为一个，这说明创建一个 ComponentName 需要指定包名和类名——这样就可唯一确定一个组件类，应用程序即可根据给定的组件类去启动特定的组件。

2. Action、Category 属性

Intent 的 Action、Category 属性都是一个普通的字符串，其中 Action 代表该 Intent 所要完成的一个抽象"动作"，而 Category 则用于为 Activity 增加额外的附加类别信息。通常 Action 属性会与 Category 属性结合使用。

Action 要完成的只是一个抽象的动作，这个动作具体由哪个组件(或是 Activity，或是 BroadcastReceiver)来完成，与 Action 这个字符串本身无关。例如，Android 提供的标准 Action：Intent. ACTION_VIEW，只表示一个抽象的查看操作，但具体查看什么、启动哪个 Activity 来查看，Intent. ACTION_VIEW 并不知道。这取决于 Activity 的<intent-filter…>配置，只要某个 Activity 的<intent-filter…/>配置中包含该 ACTION_VIEW，该 Activity 就有可能被启动。

3. Data、Type 属性

Data：数据，就像动作和类别一样，这个配置可以出现多次或一次都不出现。

例如：

```
<data android:scheme="file"/>
<data android:scheme="content"/>
<data android:mimeType="Image/png"/>
```

每个数据<data>元素可以指定一个 URI 和一个数据类型(MIME 媒体类型)。URI 由 Scheme、Authority 和 path 组成。

例如，在下面的 URI 中：

Content://com. android. provider. MyProvider/user

Scheme 是 content，Authority 为 com. android. provider. MyProvider，路径为/user。

当一个 Intent 对象中的 URI 用来和一个 IntentFiler 中的 URI 比较时，实际上比较上面

提到的 URI 各个部分。例如，如果 IntentFiler 仅指定了 scheme，所有 scheme 的 URI 和这个 IntentFilter 都匹配；如果 IntentFilter 指定了 scheme、Authority 但没有指定 path，所有相同 scheme 和 Authority 的 URI 可以匹配上，而不管它们的 path；如果 IntentFilter 指定了 scheme、Authority 和 path，只有相同 scheme、Authority 和 path 的 URI 可以匹配。当然，一个 IntentFilter 中的 path 可以包含通配符，这样只需要部分匹配即可。

数据<data>元素的类型属性指定了数据的 MIME 类型，这在 IntentFilter 里比在 URI 更为常见。Intent 对象和 IntentFilter 都可使用一个“ * ”通配符指定子类型字段，如 text/ * 或 audio/ * 表示任何匹配的子类型。

数据<data>同时比较 Intent 对象和 IntentFilter 中指定的 URI 和数据类型，匹配规则如下：

- 一个既不包含 URI 也不包含数据类型的 Intent 对象，仅在 IntentFilter 中也没有指定任何 URI 和数据类型的情况下才能通过。
- 一个包含 URI 但没有数据类型的 Intent 对象，仅在它的 URI 和一个同样没有指定数据类型的 IntentFilter 里的 URI 匹配时才能通过。这通常发生在类似于 mailto 和 tel 这样的 URI 上，它们并不是引用实际数据。
- 一个包含数据类型但不包含 URI 的 Intent 对象仅在这个 IntentFilter 中列举了同样的数据类型而且也没有指定一个 URI 的情况下才能通过。
- 一个同时包含 URI 和数据类型(或可从 URI 推断出数据类型)的 Intent 对象，如果它的类型和 IntentFilter 列举的类型相匹配，则可以通过；如果它的 URI 和这个 IntentFilter 中的一个 URI 相匹配或它有一个内容或文件 URI，而且这个 IntentFilter 仅指定了类型，那么它也能通过。

下面通过一个案例来演示 Intent 的属性。

【例 8-3】 在 Activity 使用包含项定义动作的隐式 Intent 启动另外一个 Acitivity。其具体实现步骤为：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 Intent1。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在布局文件 main1.xml 中保留一个按钮，并修改其默认属性，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/kp"
    android:orientation = "vertical" >
    <Button
        android:id = "@ + id/button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "@string/button"
        android:textColor = "@android:color/black"/>
</LinearLayout>
```

(3) 在 res\layout 目录下创建一个 main2.xml 布局文件，在文件中添加文本框控件来显示字符串，并修改默认属性。修改完成后布局代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
```

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/kp"
    android:orientation = "vertical" >
    <TextView
        android:id = "@ + id/textView"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "@string/text"
        android:textColor = "@android:color/black"
        android:textSize = "25px" />
</LinearLayout>
```

注意：在以上代码中，设置了图形的背景图为 kp，该图放置在 res\drawble-hdpi 文件夹中。

(4) 打开 src\fs.intent1 包下的 Main1Activity.java 文件，在文件中获得布局文件中的按钮控件并为其增加单击事件监听器，并在监听器中传递包含的隐式 Intent，其代码为：

```
package fs.intent1;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class Main1Activity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main1);                    //设置页面布局
        Button button = (Button) findViewById(R.id.button1);//通过 id 值获得按钮对象
        button.setOnClickListener(new View.OnClickListener()
        {
                //为按钮增加单击事件监听器
                public void onClick(View v) {
                    Intent intent = new Intent();          //创建 Intent 对象
                    intent.setAction(Intent.ACTION_VIEW);  //为 Intent 设置动作
                    startActivity(intent);                 //将 Intent 传递给 Activity
                }
            });
     }
}
```

注意：在 Main1Activity 类的代码中，并没有指定将 Intent 对象传递给哪个 Activity。

(5) 在 src\fs.intent1 包下创建一个 Main2Activity.java 文件，其具体实现操作为：

```
package fs.intent1;
import android.app.Activity;
import android.os.Bundle;
public class Main2Activity extends Activity{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main2);                    //设置页面布局
    }
}
```

(6) 双击 AndroidManifest. xml 文件，为两个 Activity 设置不同的 Intent 过滤器，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.intent1"
    android:versionCode = "1"
    android:versionName = "1.0" >
    <uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    <application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        <Activity
            android:name = "fs.intent1.Main1Activity"
            android:label = "@string/app_name" >
            <intent - filter>
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter>
        </Activity>
                <Activity
            android:name = "fs.intent1.Main2Activity"
            android:label = "@string/app_name" >
            <intent - filter>
                <action android:name = "android.intent.action.VIEW" />
                <category android:name = "android.intent.category.DEFAULT" />
            </intent - filter>
        </Activity>
    </application>
</manifest>
```

(7) 设置标题。打开 res\value\string. xml 文件，该文件用于设置界面标题，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <string name = "app_name"> Intent1 </string>
    <string name = "button">转到下一个 Activity</string>
    <string name = "text">第二个 Activity</string>
</resources>
```

(8) 启动程序后，在弹出的界面中单击“转到下一个 Activity”按钮，显示如图 8-12(a)所示的界面。单击图 8-12(a)中的 Intent1 选项，即跳转到第二个 Activity，界面如图 8-12(b)所示。

说明：由于有多种匹配 ACTION_VIEW 的方式，因此需要用户进行选择。

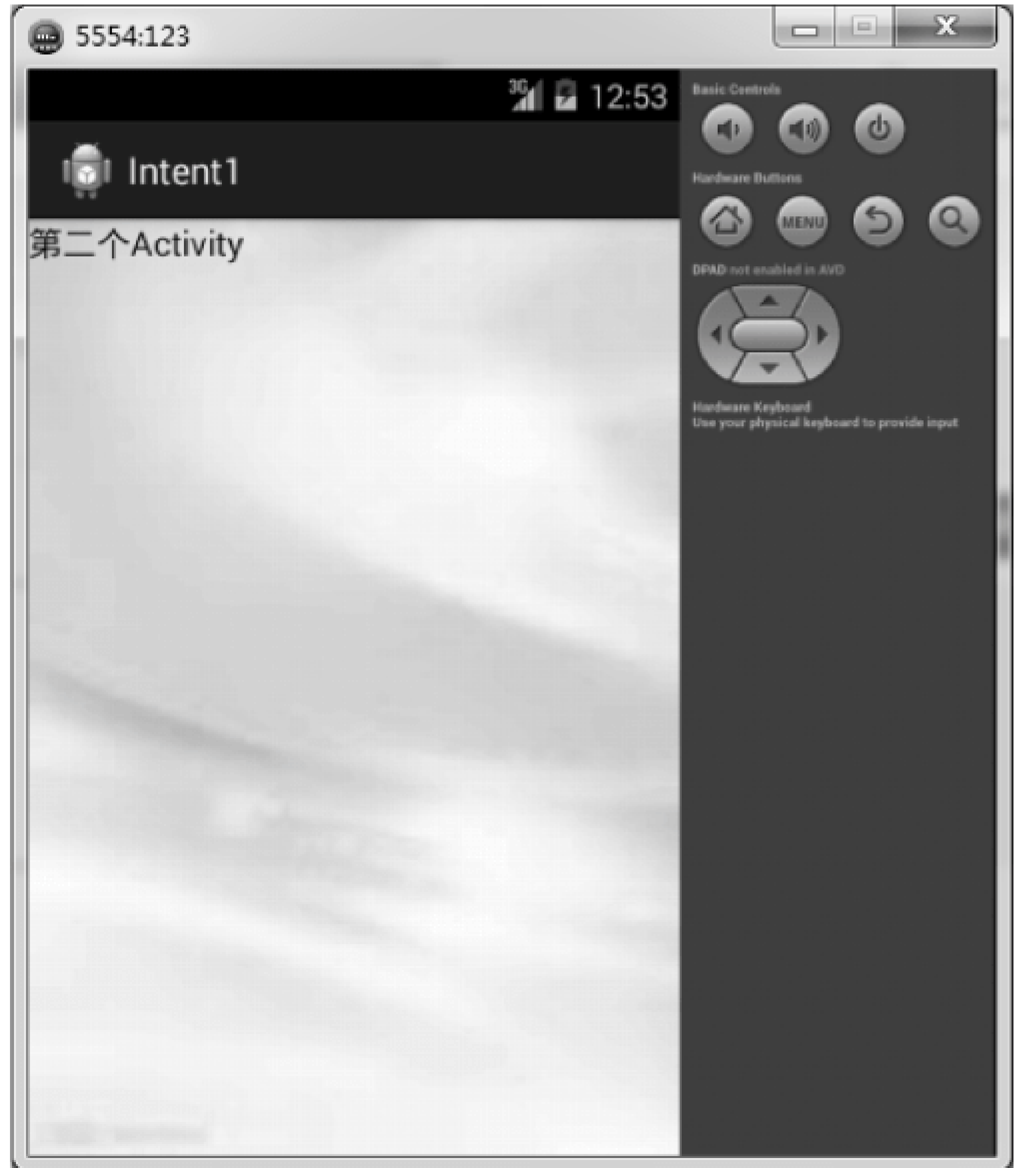
8.5.2 意图传递对象

目前所知道的意图传递对象有两种，一种是 Bundle. putSerializable(Key，Object)；另一种是 Bundle. putParcelable(Key，Object)。当然这些 Object 是有一定的条件的，前者实现了 Serializable 接口，而后者实现了 Parcelable 接口。

下面通过一个案例来演示 Intent 传递对象，其实现操作为：



(a) 选择发送方式界面



(b) 第二个Activity界面

图 8-12　Intent 使用

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 Intent_2。

(2) 打开 res\layout 目录下的 main.xml 布局文件,在文件中声明一个文本框及两个按钮控件,其代码为:

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "fill_parent"
    android:orientation = "vertical"
    android:layout_height = "fill_parent"
    android:background = "@drawable/bj">
<TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "Intent 的实例" />
<Button
        android:id = "@ + id/bn1"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "Serializable 接口"/>
<Button
        android:id = "@ + id/bn2"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "Parcelable 接口"/>
</LinearLayout>
```

(3) 新建两个类。一个为 person.java,其实现 Serializable 接口;另一个为 book.java,实现 Parcelable 接口。代码分别如下:

person.java 文件代码为:

```
package fs.intent_2;
```

```
import java.io.Serializable;
public class person implements Serializable
{
  private static final long serialversionUID = -7100;
  private String name;
  private int age;
  public String getName()
  {
   return name;
  }
  public void setName(String name)
  {
   this.name = name;
  }
  public int getAge()
  {
   return age;
  }
  public void setAge(int age)
  {
   this.age = age;
  }
}
```

book.java 文件代码为：

```
package fs.intent_2;
import android.os.Parcel;
import android.os.Parcelable;
public class book implements Parcelable
{
    private String bookname;
    private String author;
    private int publishTime;
    public String getBookname()
    {
        return bookname;
    }
    public void setBookname(String bookname)
    {
        this.bookname = bookname;
    }
    public String getAuthor()
    {
        return author;
    }
    public void setAuthor(String author)
    {
        this.author = author;
    }
    public int getPublishTime()
    {
        return publishTime;
    }
    public void setPublishTime(int publishTime)
    {
```

```
        this.publishTime = publishTime;
    }
    public int describeContents()
    {
        //TODO 自动存根法
        return 0;
    }
    public void writeToParcel(Parcel parcle, int flags)
    {
        //TODO 自动存根法
        parcle.writeString(bookname);
        parcle.writeString(author);
        parcle.writeInt(publishTime);
    }
    public static final Parcelable.Creator<book> CREATOR = new Creator<book>()
    {
        public book createFromParcel(Parcel source)
        {
            book mbook = new book();
            mbook.bookname = source.readString();
            mbook.author = source.readString();
            mbook.publishTime = source.readInt();
            return mbook;
        }
        public book[] newArray(int size)
        {
            return new book[size];
        }
    };
}
```

(4) 创建 objecttrandemo.java，并且新建两个 Activity，一个为 objecttrandemo1.java；另一个为 objecttrandemo2.java，分别用来显示 person 对象数据和 book 对象数据。代码分别如下：

objecttrandemo.java 文件代码为：

```
package fs.intent_2;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class objecttrandemo extends Activity
{
    private Button bn1,bn2;
    public final static String SER_KEY = "com.Liming.ser";
    public final static String PAR_KEY = "com.Liming.par";
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        setupView();
        bn1.setOnClickListener(new OnClickListener()
```

```
            {
                public void onClick(View v)
                {
                    //TODO 自动存根法
                    SertalizaleMethod();
                }
            });
            bn2.setOnClickListener(new OnClickListener()
            {
                public void onClick(View v)
                {
                    //TODO 自动存根法
                    PacelableMethod();
                }
            });
        }
        public void setupView()
        {
         bn1 = (Button)findViewById(R.id.bn1);
         bn2 = (Button)findViewById(R.id.bn2);
        }
        //Serializable 传递对象的方法
        public void SertalizaleMethod()
        {
            person mperson = new person();
            mperson.setName("Liming");
            mperson.setAge(20);
            Intent intent = new Intent(this,objecttrandemo1.class);
            Bundle bundle = new Bundle();
            bundle.putSerializable(SER_KEY, mperson);
            intent.putExtras(bundle);
            startActivity(intent);
        }
        //Pacelable 传递对象方法
        public void PacelableMethod()
        {
            book mbook = new book();
            mbook.setBookname("Android 实战教程");
            mbook.setAuthor("Lingming");
            mbook.setPublishTime(2012);
            Intent intent = new Intent(this,objecttrandemo2.class);
            Bundle bundle = new Bundle();
            bundle.putParcelable(PAR_KEY, mbook);
            intent.putExtras(bundle);
            startActivity(intent);
        }
        public boolean onCreateOptionsMenu(Menu menu)
        {
            getMenuInflater().inflate(R.menu.main, menu);
            return true;
        }
    }
```

objecttrandemo1.java 文件的代码为：

```
package fs.intent_2;
```

```
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
public class objecttrandemo1 extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        //TODO 自动存根法
        super.onCreate(savedInstanceState);
        TextView textview = new TextView(this);
        person p = (person)getIntent().getSerializableExtra(objecttrandemo.SER_KEY);
        textview.setText("您的名字为:" + p.getName() + "\n" + "您的年龄为:" + p.getAge());
        setContentView(textview);
    }
}
```

objecttrandemo2.java 文件的代码为：

```
package fs.intent_2;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
public class objecttrandemo2 extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        //TODO 自动存根法
        super.onCreate(savedInstanceState);
        TextView textview = new TextView(this);
        book b = (book)getIntent().getParcelableExtra(objecttrandemo.PAR_KEY);
         textview.setText("书名称为:" + b.getBookname() + "\n" + "书出版时间为:" +
b.getPublishTime() + "\n" + "书作者为:" + b.getAuthor());
        setContentView(textview);
    }
}
```

(5) 授予权限。打开 AndroidManifest.xml 文件，其代码为：

```
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.intent_2"
    android:versionCode = "1"
    android:versionName = "1.0" >
    <uses-sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "15" />
    <application
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        <activity
            android:name = ".objecttrandemo"
            android:label = "@string/title_activity_main" >
            <intent-filter>
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
```

```
            </intent-filter>
        </activity>
         <activity android:name=".objecttrandemo1"></activity>
        <activity android:name=".objecttrandemo2"></activity>
    </application>
</manifest>
```

运行程序，初始界面如图 8-13(a)所示。单击“Serializable 接口”按钮，效果如图 8-13(b)所示；单击“Parcelable 接口”按钮，效果如图 8-13(c)所示。



(a) Intent传递对象初始界面　(b) Serializable接口界面　(c) Parcelable接口界面

图 8-13　Intent 传递对象

在 Activity 之间传递对象时有两个选择：一个为 Parcelable；另一个为 Serializable。但究竟该何时使用其中的一个呢？很多人不得而知，所以混用和滥用的情况就出现了。

注意：

(1) 在使用内存时，Parcelable 类比 Serializable 性能高，所以推荐使用 Parcelable 类。

(2) Serializable 在序列化时会产生大量的临时变量，从而引起频繁的垃圾回收(Games Convention，GC)。

(3) Parcelable 不能使用在要将数据存储在磁盘上的情况，因为 Parcelable 不能在外界有变化的情况下很好地保证数据的持续性。尽管 Serializable 效率低，也不提倡用，但在这种情况下，还是建议使用 Serializable。

第 9 章　Android 通信

手机作为一种通信终端,伴随着网络的升级而不断的升级换代。1995 年 1G 问世,手机只能进行基本的语音通信,1996—1997 年 2G(GSM、CDMA)及其后的 GPRS、EDGE 等技术的快速发展,手机开始逐渐增加了数据服务功能。2009 年开始,3G 在全世界开始大规模布置以及苹果公司创造性开发新型苹果手机。手机慢慢地变成互联网的终端,从而带动了一个新的时代——移动互联网时代。现代手机通常都支持这些常用网络设备,如 WIFI、NF、蓝牙等。

Android 是由互联网巨头 Google 公司带头开发的,因此对网络功能的支持是必不可少的。Google 公司的应用层采用的是 Java 语言。所以 Java 支持的网络编程方式 Android 都支持,同时 Google 公司还引入了 Apache 的 HTTP 扩展包。另外,Android 针对 WIFI 和 NFC,分别提供的单独的开发 API。

9.1　通 信 方 式

Android 平台有三种网络接口可以使用,它们分别是 java. net. *(标准 Java 接口)、Org. apache 接口(Apache 接口)和 Android. net. *(Android 网络接口)。

其中,

- java. net. *(标准 Java 接口),提供包括流和数据包套接字、Internet 协议、常用 HTTP 处理。该包是一个功能很全面的网络通信包,方便有经验的 Java 开发人员直接使用。
- Org. apache(Apache 接口):为 HTTP 通信提供了高效、精确、功能丰富的工具包支持。
- Android. net. *(Android 网络接口):提供了网络访问的 SOCKET、URI 类似及和 WiFi 相关的类,并且提供了网络状态监视管理等接口。

有了这些工具包的支持,在 Android 中具体使用如下网络编程的方式为:

(1) 针对 TCP/IP 的 Socket、ServerSocket。

(2) 针对 UDP 的 DatagramSocket、DatagramPackage。

(3) 针对直接 URL 的 HttpURLConnection。

(4) Google 公司集成了 Apache HTTP 客户端,可使用 HTTP 进行网络编程。

(5) 使用 Web Service 进行网络编程。

(6) 直接使用 WebView 视图组件显示网页。

其中,方式(1)和(2)都为 Socket 通信方式,方式(3)、(4)、(5)为 HTTP 通信方式,而方式(6)为 Android 提供的网页浏览控件。

9.2 TCP 通信

java.net.* 提供与联网有关的类，包括流、数据包套接字(Socket)、Internet 协议、常见 HTTP 处理等。例如，创建 URL，以及 URLConnection/HttpURLConnection 对象、设置链接参数、链接到服务器、向服务器写数据、从服务器读取数据等通信。

应用程序通过套接字进行通信，可以使用 UDP 协议或使用 TCP 协议。当客户端和服务器端的协议相对应时，客户端使用 TCP，那么服务器端使用 TCP。Socket 通信模型如图 9-1 所示。

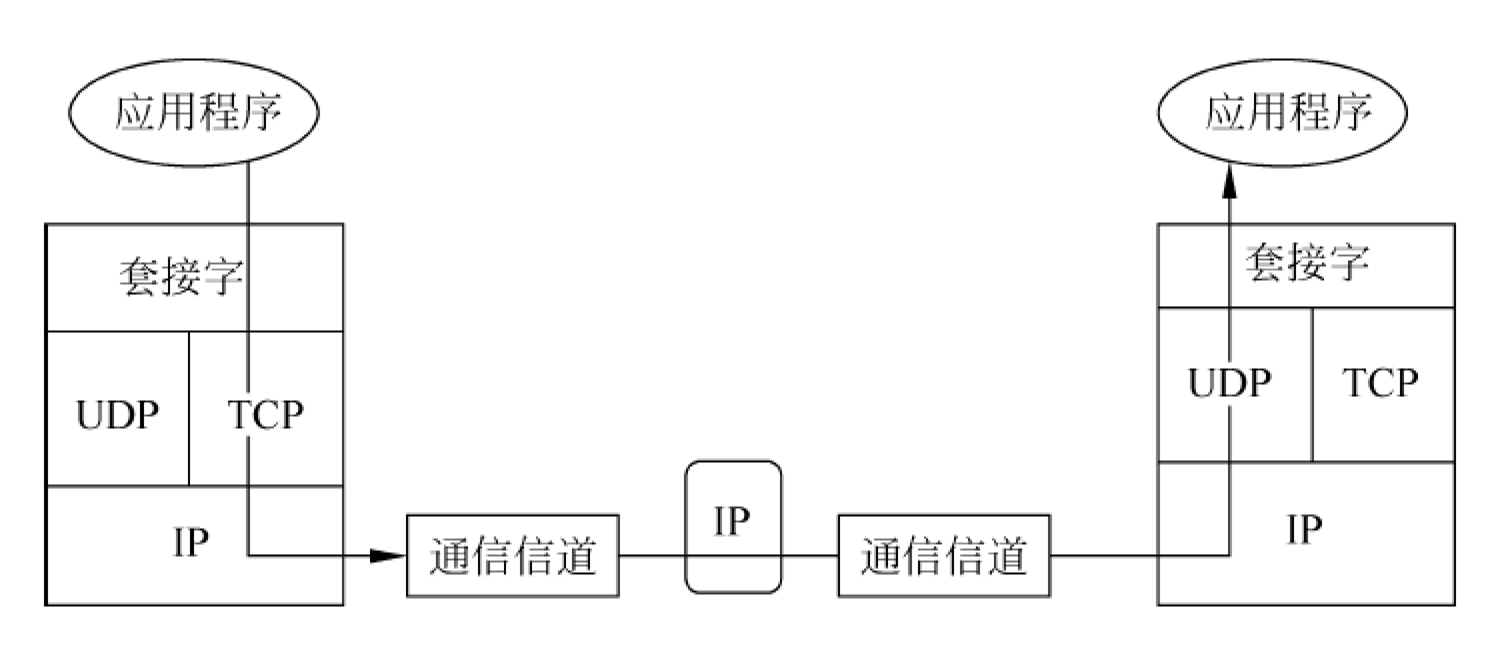


图 9-1　Socket 通信模型

在本节中，可以使用 TCP 通信方式来实现 Android 远程控制 PC 的效果，主要分为 PC 服务器端以及 Android 控制端。TCP 通信流程如图 9-2 所示。

该模式下的通信时，服务器端先启动侦听服务，等待客户端的连接。客户端连接到服务端后发送请求到服务器端，服务器端处理请求并做出相应的应答，实现通信。

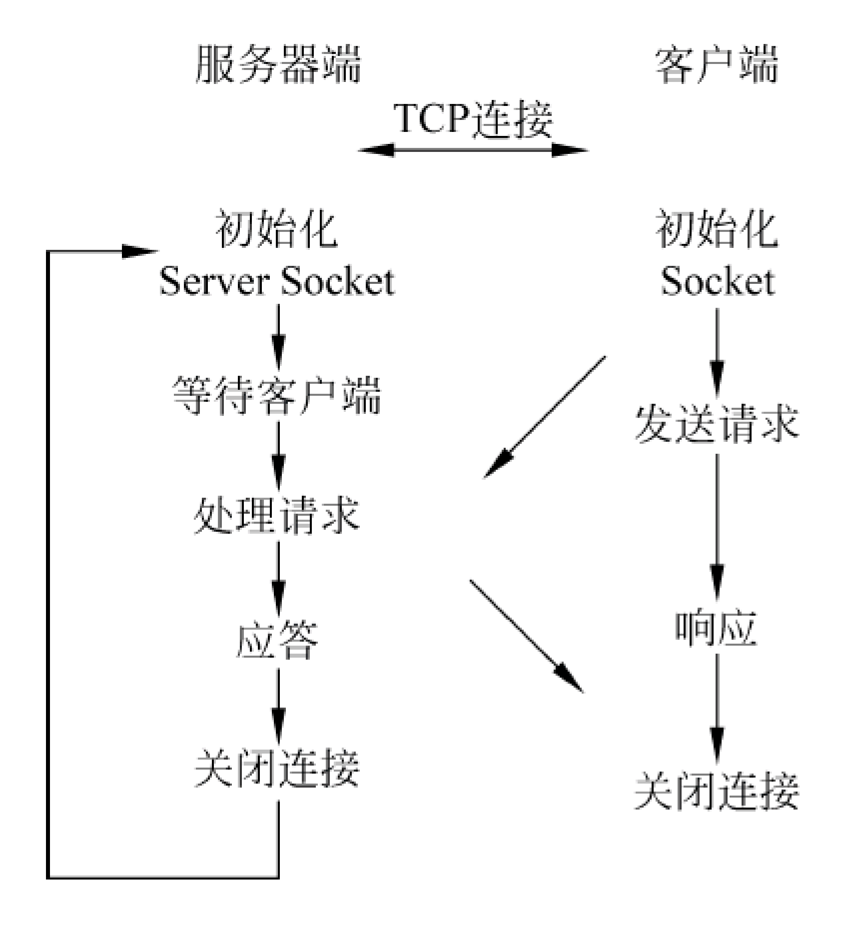


图 9-2　TCP 通信流程

9.2.1 PC 服务器端

PC 的 IP 相对固定，作为服务器端运行，需要完成服务器端的 TCP 通信流程以及关闭 PC 的操作。值得注意的是，由于服务器端是运行 PC 的程序，所以需要创建的是一个 Java 的标准项目，不再是 Android 项目。

从 TCP 的通信流程可以看出，在服务器端需要完成以下 4 个步骤。

(1) 创建服务器端套接字并绑定到一个端口。在 Java 标准接口中，提供了两个类 ServerSocket 和 Socket，分别用来表示服务器端和客户端。服务器端的 ServerSocket 有如下几种构造函数。

```
ServerSocket()
ServerSocket(int aport)
ServerSocket(int aport, int backlog)
ServerSocket(int aport, int backlog, InetAddress localAddr)
```

其中，参数 aport 指定服务器要绑定的端口即为服务器要监听的端口，参数 backlog 指定客户连接请求队列的长度，参数 logcalAddr 指定服务器要绑定的 IP 地址。一般来说端口号 0～923 为系统预留的，使用的端口最好大于 924。例如，创建一个监听端口号为 3344 的服务

套接字,其代码为:

```
ServerSocket serversocket = new ServerSocket(3344);
```

(2) 套接字设置监听模式等待连接请求。创建服务套接字后,接下来就监听端口等待客户端的连接,使用 ServerSocket 类的方法为:

```
Socket accept()
```

该方法为一个阻塞方法,调用该方法后将一直监听端口等待客户端的请求,直到有客户端连接到该端口后,才会返回一个对应客户端的 Socket,继续执行之后的代码。

(3) 接受连接请求后进行通信。Socket 连接建立后,服务器端和和客户端通过 Socket 的输入输出流来读写数据,实现通信的功能。Socket 提供的方法为:

```
InputStream getInputStream()
OutputStream getOutputStream()
```

分别返回用于读取数据的 InputStream 类对象和用于写入数据的 outputStream 类对象。为了方便读写数据,可以使用流 DataInputStream 和 DataOutputStream 类;对于文本流对象,可以使用 InputStreamReader 和 OutputStreamReader 类。以使用 DataInputStream 类读取输入请求为例,其代码为:

```
DataInputStream data_input = new DataInputStream(Client_socket,getInputStream());
String msg = data_input,readUTF();
```

(4) 关闭该 Socket 返回,等待下一个连接请求。通信完成后,需要将输入输出流以及 Socket 关闭,以主动释放不再使用的资源。

TCP 的通信方式如图 9-3 所示。

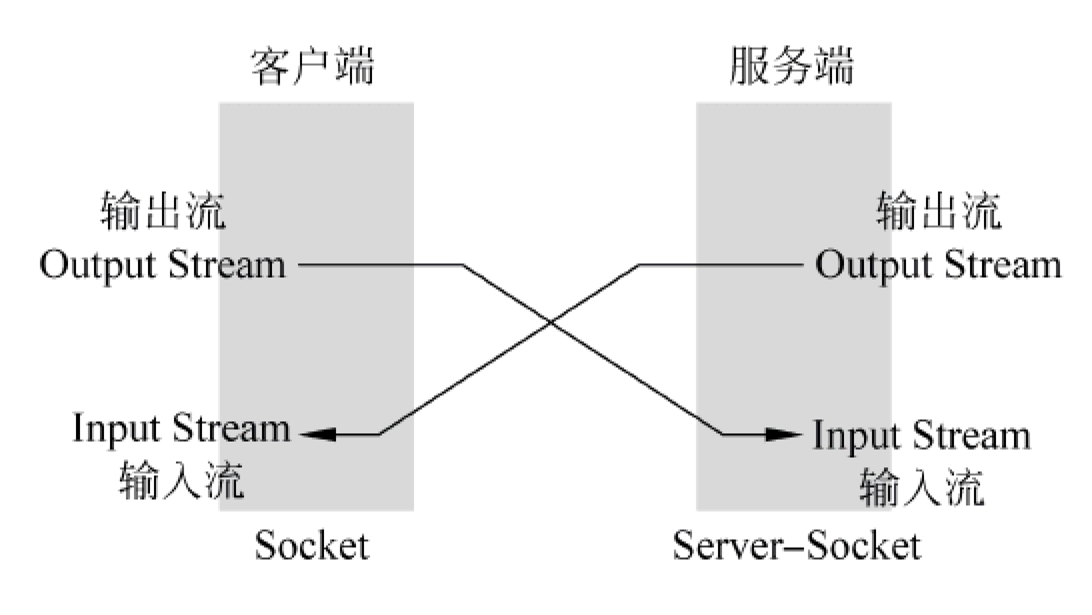


图 9-3 TCP 通信方式

9.2.2 TCP 通信经典案例

熟悉了整个通信过程以及关键点,实现在 PC 上运行的服务器端,对客户端输入的命令进行判断,执行不同命令对应的操作。

【例 9-1】 本战例实现 Android 客户端 Socket 连接 PC 服务器端。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 li9_1Socket_PC。

(2) 打开 res\layout 目录下的 main.xml 文件,在文件中布局一个 TextView 控件、一个 EditText 控件及一个 Button 控件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
     android:background = "@drawable/kp" >
     <TextView
         android:text = "TCP 通信"
         android:id = "@ + id/TextView01"
         android:layout_width = "wrap_content"
         android:layout_height = "wrap_content"/>
```

```
    <EditText
        android:id = "@ + id/EditText01"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/Button01"
        android:layout_below = "@ + id/Button01"
        android:layout_marginTop = "36dp"
        android:ems = "10"
        android:text = "输入流"/>
     <Button
        android:id = "@ + id/Button01"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/TextView01"
        android:layout_below = "@ + id/TextView01"
        android:layout_marginTop = "20dp"
        android:text = "关闭" />
</LinearLayout>
```

(3) 打开 src\fs.li9_1socket_pc 包下的 MainActivity.java 文件，在文件中实现创建 Socket 及实现 Socket 连接，其代码为：

```
package fs.li9_1socket_pc;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.Socket;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
public class MainActivity extends Activity{
    private TextView mTextView = null;
    private EditText mEditText = null;
    private Button mButton = null;
    /** 第一次调用 activity. */
        @Override
        public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mButton = (Button) findViewById(R.id.Button01);
        mTextView = (TextView) findViewById(R.id.TextView01);
        mEditText = (EditText) findViewById(R.id.EditText01);
      //登录
     mButton.setOnClickListener(new OnClickListener(){
         public void onClick(View v){
             Socket socket = null;
             String message = mEditText.getText().toString() + "\r\n";
               try {
                   //创建 Socket
```

```
                        socket = new Socket("192.168.1.100",5554);
                        //查看本机 IP,每次开机都不同
                        //向服务器发送消息
                        PrintWriter out = new PrintWriter(new BufferedWriter(new OutputStreamWriter
(socket.getOutputStream())),true);
                        out.println(message);
                       //接收来自服务器的消息
                       BufferedReader br = new BufferedReader ( new InputStreamReader ( socket.
getInputStream()));
                        String msg = br.readLine();
                        if (msg!= null) {
                            mTextView.setText(msg);
                        } else {
                           mTextView.setText("数据错误!");
                        }
                        //关闭流
                        out.close();
                        br.close();
                        //关闭 Socket
                        socket.close();
                    } catch (Exception e){
                        //异常处理
                        Log.e("", e.toString());
                    }
                }
            });
        }
    }
```

(4) 在 src\fs.li9_1socket_pc 包下创建一个 Server.java 文件,用于实现服务器端连接,其代码为:

```
package fs.li8_1socket_pc;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
public class Server implements Runnable {
    public void run() {
        try {
            //创建 ServerSocket
         ServerSocket serverSocket = new ServerSocket(5554);
            while (true) {
                //接受客户端请求
                Socket client = serverSocket.accept();
                System.out.println("accept");
                try {
                    //接收客户端消息
                    BufferedReader in = new BufferedReader(
                            new InputStreamReader(client.getInputStream()));
                    String str = in.readLine();
```

```
                        System.out.println("read:" + str);
                        //向服务器发送消息
                        PrintWriter out = new PrintWriter(new BufferedWriter(
                                new OutputStreamWriter(client.getOutputStream())),true);
                        out.println("server message");
                       //关闭流
                        out.close();
                        in.close();
                    } catch (Exception e) {
                        System.out.println(e.getMessage());
                        e.printStackTrace();
                    } finally {
                        //关闭
                        client.close();
                        System.out.println("close");
                    }
                }
            } catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }
        //main 函数,开启服务器
        public static void main(String a[]) {
            Thread desktopServerThread = new Thread(new Server());
            desktopServerThread.start();
        }
    }
```

(5) 打开 AndriodManifest.xml 文件,代码修改为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.li9_1socket_pc"
    android:versionCode = "1"
    android:versionName = "1.0" >
    <uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    <application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        <activity
            android:name = "fs.li9_1socket_pc.MainActivity"
            android:label = "@string/app_name" >
            <intent - filter>
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter>
        </activity>
    </application>
   <uses - permission android:name = "android.permission.INTERNET"/>
</manifest>
```

运行程序，效果如图 9-4 所示。



图 9-4　TCP 通信

【例 9-2】 本战例实现 Android 客户端连接 PC 服务器端（Socket 连接）。其具体实现步骤为：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 li9_2PC_Socket。

（2）res\layout 目录下的 main.xml 文件的代码与例 9-1 相类似。

（3）打开 src\fs.li9_2pc_socket 包下的 MainActivity.java 文件，实现客户端连接 PC 服务器的功能，其代码为：

```
package fs.li8_2pc_socket;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import android.app.Activity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;
import android.view.View;
import android.view.View.OnClickListener;
public class MainActivity extends Activity implements OnClickListener{
    Socket socket;
    DataInputStream dis;
    DataOutputStream dos;
    private TextView mTextView1;
    private Button Button01;
    /** 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mTextView1 = (TextView) findViewById(R.id.rec);
        Button01 = (Button) findViewById(R.id.Button01);
        Button01.setOnClickListener(this);
        Client("127.0.0.1");
        }
    public void Client(String IP) {
        try {
            //创建一个 Socket 流连接到目标主机
            socket = new Socket(IP, 10000);
            //输入流读出数据输出流写数据
            dis = new DataInputStream(socket.getInputStream());
            dos = new DataOutputStream(socket.getOutputStream());
            } catch (IOException ioe) {
                ioe.printStackTrace();
                }
        }
    //写数据到 Socket
    public void WriteInt(int i) {
        try {
```

```
                dos.writeInt(i);
                dos.flush();
                } catch (IOException ioe) {
                    ioe.printStackTrace();
                    }
            }
        public void WriteString(String str){
            StringBuffer message = new StringBuffer();
            message.append(str);
            byte[ ] b = new byte[6];
            try {
                dos.write(12);
                } catch (IOException e) {
                    //TODO 自动存根法
                    e.printStackTrace();
                    }
            }
        //显示从 Socket 返回的数据
        public void ReadInt() {
            try {
                mTextView1.setText(dis.readInt());
                System.out.println(dis.readInt());
                } catch (IOException ioe) {
                    ioe.printStackTrace();
                    }
            }
        @Override
        public void onClick(View v) {
            //TODO 自动存根法
            WriteInt(25);
            }
        }
```

(4) 在 src\fs.li9_2pc_socket 包下新建一个 Server.java 文件，实现服务端，其代码为：

```
package fs.li9_2pc_socket;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
public class Server {
    ServerSocket serversocket;
    Socket socket;
    DataInputStream dis;
    DataOutputStream dos;
    public Server() {
        try {
            serversocket = new ServerSocket(1900);
            System.out.println("等待 Client 连接");
            //侦听套接字上的连接
            socket = serversocket.accept();
            System.out.println("Client 已连接");
            dis = new DataInputStream(socket.getInputStream());
            dos = new DataOutputStream(socket.getOutputStream());
            } catch (IOException ioe) {
```

```
            ioe.printStackTrace();
            }
        }
        public void WriteInt(int i){
            try {
                //向 Socket 的输出流写入 Int(32 位 integer)数据
                //如果是 4 字节,则从高到低写入
                dos.writeInt(i);
                dos.flush();
                } catch (IOException ioe) {
                    ioe.printStackTrace();
                    }
                }
                //从 Socket 的输入流读出 int(32 位 integer)数据
                public void ReadInt() {
                    try {
                        System.out.println(dis.readInt());
                        } catch (IOException ioe) {
                            ioe.printStackTrace();
                            }
                    }
                    public void readString(){
                        try{
                            System.out.println(dis.readUTF());
                            }catch(IOException e){
                                e.printStackTrace();
                                }
                        }
                    public static void main(String args[]){
                    Server theServer = new Server();
                    //接收服务器端已经成功,获取客户端发送来的
                    theServer.ReadInt();
                    theServer.readString();
                    theServer.WriteInt(10);
            }
}
```

9.3 UDP 通信

UDP 通信方式是无连接的 Socket 通信,所以不需要像 TCP 那样先建立连接再发送数据,可以直接对目标地址发送数据。这样的方式更加快速、高效,但是不能保证数据能够完全到达目标端。

9.3.1 UDP 通信概述

UDP(User Datagram Protocol,用户数据包协议)是 OSI 参考模型中一种无连接的传输层协议,提供面向事务的简单不可靠信息传送服务。它是 IETF RFC 768,是 UDP 的正式规范。在网络中它与 TCP 协议一样用于处理数据包。在 OSI 模型中,它在第 4 层——传输层,处于 IP 协议的上一层。UDP 有不提供数据报分组、组装和不能对数据包的排序的缺点,也就是说,当报文发送之后,是无法得知其是否安全到达的。UDP 用来支持那些需要在计算机之间传输数据的网络应用。包括网络视频会议系统在内的众多的客户/服务器模式的网络应用

都需要使用 UDP 协议。UDP 协议从问世至今已经被使用了很多年，虽然其最初的光彩已经被一些类似协议所掩盖，但是即使是在今天，UDP 仍然不失为一项非常实用和可行的网络传输层协议。与所熟知的 TCP(传输控制协议)协议一样，UDP 协议直接位于 IP(网际协议)协议的顶层。根据 OSI(开放系统互连)参考模型，UDP 和 TCP 都属于传输层协议。

UDP 协议的主要作用是将网络数据流量压缩成数据报的形式。一个典型的数据报就是一个二进制数据的传输单位。每一个数据报的前 8 字节用来包含报头信息，剩余字节则用来包含具体的传输数据。TCP 和 UDP 在 Android 中的使用和在 Java 里是完全一样的。

9.3.2 UDP 通信流程

UDP 通信相对 TCP 通信而言比较简单，不需要事先建立连接。只需要创建一个接收和发送的套接字即可实现数据处理和发送，UDP 的通信流程如图 9-5 所示。

UDP 通信同样分为服务器端和客户端两部分。

在 Android 的服务器端，主要用于开启端口、等待客户端的数据输入和应答。在服务器端实现需要如下几个步骤。

(1) 创建套接字并绑定到一个端口。在 UDP 中使用套接字 DatagramSocket 来表示数据的接收站和发送站。常用的构造函数有：

```
DatagramSocket()
DatagramSocket(int aPort)
DatagramSocket(int aPort,InetAddress addr)
DatagramSocket(SocketAddress localAdd)
```

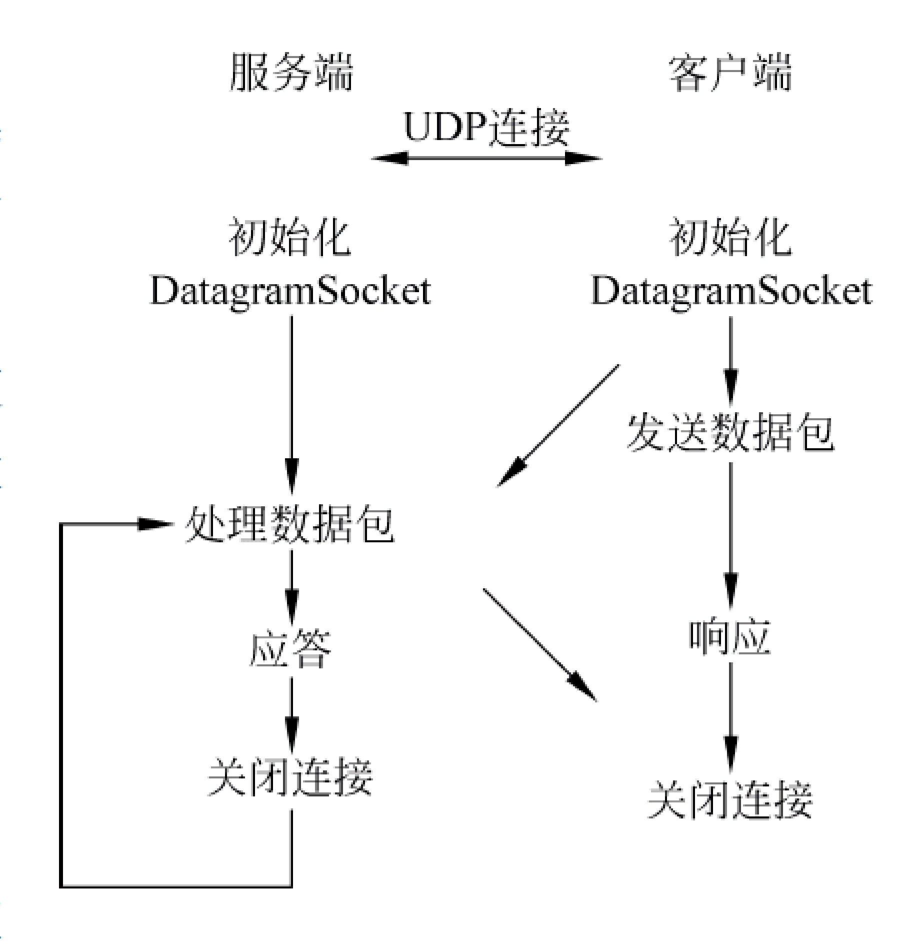


图 9-5 UDP 通信流程图

其中，aPort 为本地绑定的端口号；InetAddress 为指定的地址；SocketAddress 为表明绑定到特定的套接字地址。例如，创建一个监听端口号为 5000 的 UDP 套接字，实现代码为：

```
DatagramSocket = new DatagramSocket(5000);
```

(2) 接收数据。有了套接字后，即可直接使用其来接收数据，使用 DatagramSocket 的实现方法为：

```
receive(DatagramPack pack)
```

其中，参数 pack 为 DatagramPacket 类型，表示存放数据的数据包。

(3) 处理数据。无论是发送还是接收的数据都以 DatagramPack 类型表示，处理数据前必须构造此类，但是对于接收数据包和发送数据包是有区别的。常用接收数据构造函数为：

```
DatagramPack(byte[] data,int length)
```

其中，参数 data 为接收的数据；length 为数据的长度。例如，创建一个可以存放 1024 字节数据的接收数据包，其代码为：

```
byte buf[] = new byte[1024];
DatagramPack dp = new DatagramPack(buf,1024);
```

常用的发送数据构造函数有：

```
DatagramPack(byte[] data,int length,InetAddress host,int port)
DatagramPack(byte[] data,int length,SocketAddress sockAddr)
```

其中，参数 data 为发送的数据；length 为数据的长度；InetAddress 为发送到的目标地址；port 为发送到的目标端口；SocketAddress 为发送到的指定套接字地址。在 DatagramPack 类中可以获取该包发送地址的 IP 地址、端口、套接字地址以及数据内容，分别使用如下方法。

```
InetAddress getAddress()
Int getPort()
SocketAddress getSocketAddress()
byte[] getData()
```

对于 Android 的客户端而言，在 UDP 中，发送数据和接收数据的流程类似，都是通过套接字 DatagramSocket 发送或接收数据 DatagramPack。实现步骤有：

(1) 创建套接字 DatagramSocket，和接收端完全一致。

(2) 发送数据 DatagramPack。

在发送端，数据包为发送数据包，必须指定数据包发送到的目标地址和端口，使用 DatagramPack 的发送构造数据包。例如，数据包目标地址为 IP 值，端口为 5000 的数据，实现代码为：

```
DatagramPack dp = new DatagramPack(buf,buf.length,InetAddress.getByName(ip),5000);
```

数据构造好后，使用 DatagramSocket 的发送数据方法为：

```
send(DatagramPack pack)
```

9.3.3 UDP 通信经典案例

下面通过一个案例来演示 UDP 的通信。

【例 9-3】 下面利用 UDP 通信向指定 IP 发送消息“Android+i(动态自增值)”功能。其具体实现操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li9_3UDP_Communicate。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 ScrollView 控件实现自增，一个 TextView 控件实现数据的显示，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical" android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
  <ScrollView
      android:layout_width = "fill_parent"
      android:layout_height = "fill_parent">
<TextView
    android:id = "@ + id/mainTextView"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
        </TextView>
    </ScrollView>
</LinearLayout>
```

(3) 打开 src\fs.li9_3udp_ communicate 包下的 MainActivity.java 文件，在文件中实现发送端，其代码为：

```
package fs.li9_3udp_communicate;
import java.io.IOException;
```

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.net.UnknownHostException;
public class MainActivity {
    public static void main(String[] args) {
        try {
            //定义要发送的字符串并转为 byte 数组
            byte[] buffer = "Hello".getBytes();
            int messageCount = 0;
            //目标 IP 地址
            InetAddress wiFiDestIP = InetAddress.getByName("192.168.43.1");
            //定义 UDP 数据包,需要指定目标地址及端口号
            DatagramPacket packet = new DatagramPacket(buffer,buffer.length,wiFiDestIP,54321);
            //发送数据
            DatagramSocket sendSocket = new DatagramSocket();
            StringBuffer dataString = new StringBuffer();
            while(true){
                sendSocket.send(packet);
                System.out.println("发送数据:" + new String(packet.getData()));
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    //TODO 自动存根法
                    e.printStackTrace();
                }
                messageCount++;
                buffer = ("Android" + messageCount).getBytes();
                packet.setData(buffer,0,buffer.length);
            }
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (SocketException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

(4) 在 src\fs.li9_3udp_communicate 包下创建一个 UDPR.java 文件,该文件用于实现接收端,其代码为:

```
package fs.li9_3udp_communicate;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.widget.TextView;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
```

```java
import java.net.SocketOptions;
import java.net.UnknownHostException;
public class UDPR extends Activity {
    /** 第一次调用活动. */
    TextView mainTextView;
    Thread mReceiveThread;
    DatagramSocket server;
    int mMessageCountInt = 0;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mainTextView = (TextView) findViewById(R.id.mainTextView);
        //定义 UDP 监听
        try {
            server = new DatagramSocket(54321);
        } catch (SocketException e) {
            //TODO 自动存根法
            e.printStackTrace();
        }
        Log.e("myLog","activity run");
        mReceiveThread = new Thread(updateThread);
        mReceiveThread.start();
        mainTextView.append("开始接收数据: \n");
    }
    @Override
    protected void onResume() {
        super.onResume();
        if(!mReceiveThread.isAlive()){
            if(server.isClosed()){
                Log.e("myLog","Resume thread");
                //定义 UDP 监听
                try {
                    server = new DatagramSocket(54321);
                } catch (SocketException e) {
                    //TODO 自动存根法
                    e.printStackTrace();
                }
            }
            mReceiveThread = new Thread(updateThread);
            mReceiveThread.start();
        }
    }
    @Override
    protected void onPause() {
        super.onPause();
        server.close();
    }
    //使用匿名内部类来复写 Handler 当中的 handlerMessage()方法
    final Handler updateBarHandler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            Log.e("myLog","handleMessage");
            //每次最多显示 30 条消息
            if(mMessageCountInt++>=30){
```

```
                mainTextView.setText("");
                mMessageCountInt = 0;
            }
            mainTextView.append((msg.getData()).getString("data"));
        }
    };
    //线程类,该类使用匿名内部类的方式进行声明
    Runnable updateThread = new Runnable() {
        public void run() {
            //TODO 自动存根法
             Log.e("LogCat","执行 run");
            //得到一个消息对象,Message 类是 Android 系统提供的
            Message msg = new Message();
            Bundle b = new Bundle();
                try {
                    //定义缓冲区
                    byte[] buffer = new byte[1024];
                    //定义接收数据包
                    DatagramPacket packet = new DatagramPacket(buffer,
                            buffer.length);
                    while (true) {
                        msg = updateBarHandler.obtainMessage();
                        //接收数据
                        server.receive(packet);
                        //判断是否收到数据,然后输出字符串
                        if (packet.getLength() > 0) {
                            String str = new String(buffer, 0, packet
                                    .getLength());
                            b.putString("data", str + "\n");
                            msg.setData(b);
                            //将 Message 对象加入到消息队列当中
                            updateBarHandler.sendMessage(msg);
                            Log.e("LogCat", "发送消息");
                        }
                    }
                } catch (SocketException e) {
                    Log.e("DEBUG_TAG", e.toString());
                } catch (IOException e) {
                    Log.e("DEBUG_TAG", e.toString());
                }
        }
    };
}
```

(5) 打开 AndroidManifest.xml 文件,在文件中设置通信权限,其代码为:

```
⋮
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />
<uses-permission android:name="android.permission.INTERNET" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
⋮
```

运行程序，效果如图 9-6 所示。由图 9-6 可以看出，该程序被终止。

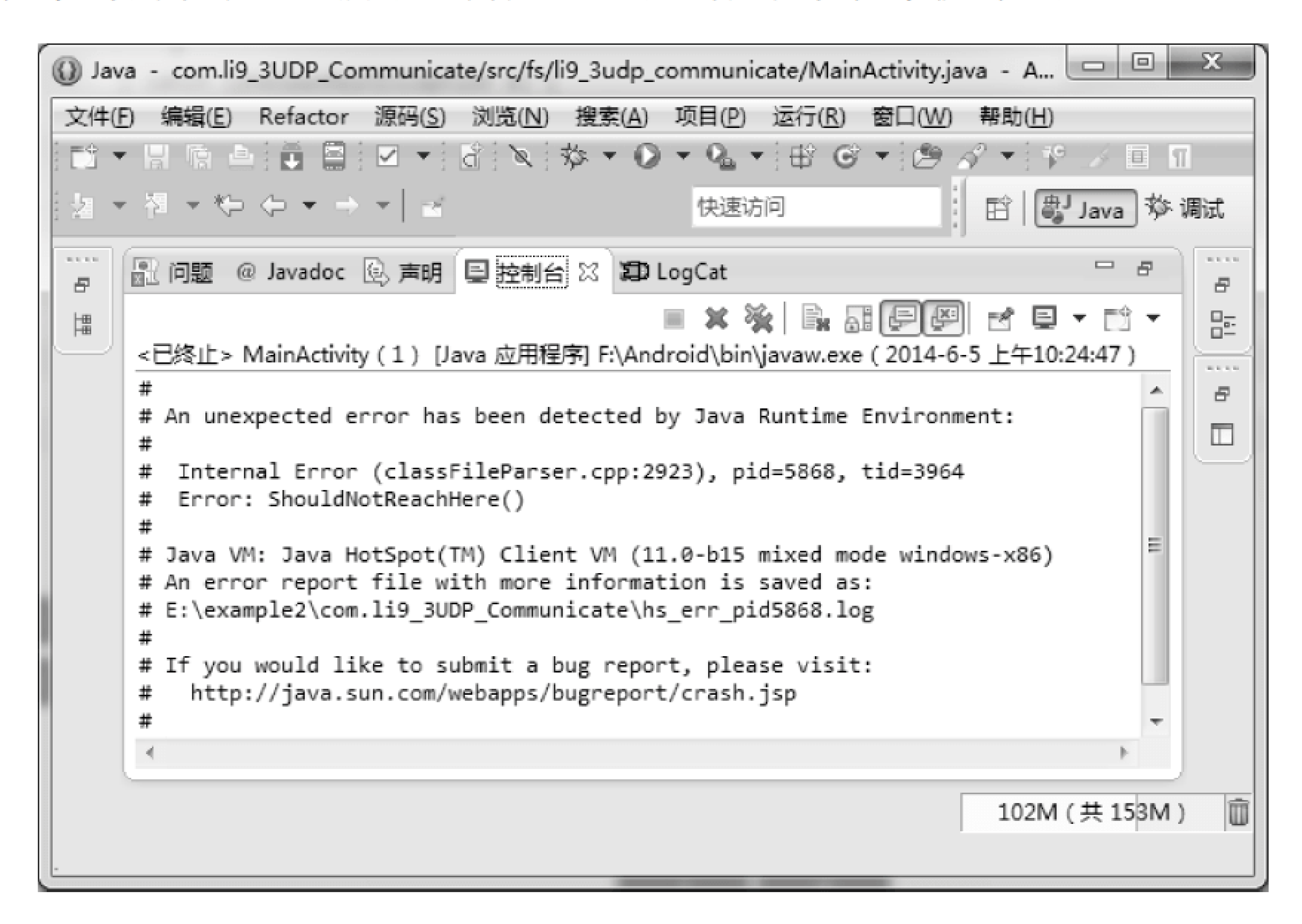


图 9-6 输出信息

9.4 HTTP 通信

众所周知，Android 与服务器通信通常采用 HTTP 通信方式和 Socket 通信方式，而 HTTP 通信方式又分 get 和 post 两种方式。

9.4.1 HTTP 通信概述

HTTP (Hypertext Transfer Protocol)是 Web 联网的基础，也是手机联网常用的协议之一，HTTP 协议是建立在 TCP 协议之上的一种协议。

HTTP 连接最显著的特点是客户端发送的每次请求都需要服务器回送响应，在请求结束后，会主动释放连接。从建立连接到关闭连接的过程称为“一次连接”。在 HTTP 1.0 中，客户端的每次请求都要求建立一次单独的连接，在处理完本次请求后，就自动释放连接。在 HTTP 1.1 中，可以在一次连接中处理多个请求，并且多个请求可以重叠进行，不需要等待一个请求结束后再发送下一个请求。

由于 HTTP 在每次请求结束后都会主动释放连接，因此 HTTP 连接是一种“短连接”、“无状态”，要保持客户端程序的在线状态，需要不断地向服务器发起连接请求。通常的做法是即使不需要获得任何数据，客户端也保持每隔一段固定的时间向服务器发送一次“保持连接”的请求，服务器在收到该请求后对客户端进行回复，表明知道客户端“在线”。若服务器长时间无法收到客户端的请求，则认为客户端“下线”，若客户端长时间无法收到服务器的回复，则认为网络已经断开。

基于 HTTP 1.0 协议的客户端在每次向服务器发出请求后，服务器就会向客户端返回响应消息，在确认客户端已经收到响应消息后，服务端就会关闭网络连接。在这个数据传输过程中，并不保存任何历史信息和状态信息，因此，HTTP 协议也被认为是无状态的协议。

HTTP 1.1 和 HTTP 1.0 相比较而言，最大的区别就是增加了持久连接支持。当客户端使用 HTTP 1.1 协议连接到服务器后，服务器就将关闭客户端连接的主动权交还给客户端；也就是说，只要不调用 Socket 类的 close 方法关闭网络连接，就可以继续向服务器发送 HTTP 请求。

HTTP 连接使用的是“请求—响应”的方式，不仅在请求时需要先建立连接，而且需要客户端向服务器发出请求后，服务器端才能回复数据。而 Socket 连接在双方建立起连接后就可以直接进行数据的传输。

1. HTTP 协议的特点

HTTP 协议具有如下几个特点：

- 支持 B/S 及 C/S 模式。
- 简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有 GET、HEAD、POST。
- 灵活：HTTP 允许传输任意类型的数据对象。正在传输的类型由 Content-Type 加以标记。
- 无状态：HTTP 协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。

2. HTTP 协议请求方法

HTTP 协议的请求方法主要有：

- GET 请求获取 Request-URI 所标识的资源；
- POST 在 Request-URI 所标识的资源后附加新的数据；
- HEAD 请求获取由 Request-URI 所标识的资源的响应消息报头；
- PUT 请求服务器存储一个资源，并用 Request-URI 作为其标识；
- DELETE 请求服务器删除 Request-URI 所标识的资源；
- TRACE 请求服务器回送收到的请求信息，主要用于测试或诊断；
- CONNECT 保留将来使用；
- OPTIONS 请求查询服务器的性能，或查询与资源相关的选项和需求。

3. HttpURLConnection 接口

首先需要明确的是，HTTP 通信中的 POST 和 GET 请求方式的不同。GET 可以获得静态页面，也可以把参数放在 URL 字符串后面，传递给服务器。POST 方法的参数是放在 HTTP 请求中。因此，在编程之前，应当首先明确使用的请求方法，然后再根据所使用的方式选择相应的编程方式。

HttpURLConnection 是继承于 URLConnection 类，两者都是抽象类。其对象主要通过 URL 的 openConnection 方法获得。创建方法如下代码所示：

```
URL url = new URL("http://www.51cto.com/index.jsp?par = 123456");
HttpURLConnection urlConn = (HttpURLConnection)url.openConnection();
```

通过以下方法可以对请求的属性进行一些设置，具体如下：

```
//设置输入和输出流
urlConn.setDoOutput(true);
urlConn.setDoInput(true);
//设置请求方式为 POST
```

```
urlConn.setRequestMethod("POST");
//POST 请求不能使用缓存
urlConn.setUseCaches(false);
//关闭连接
urlConn.disConnection();
```

HttpURLConnection 默认使用 GET 方式。例如：

```
//使用 HttpURLConnection 打开连接
HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
//得到读取的内容(流)
InputStreamReader in = new InputStreamReader(urlConn.getInputStream());
//为输出创建 BufferedReader
BufferedReader buffer = new BufferedReader(in);
String inputLine = null;
//使用循环来读取获得的数据
while (((inputLine = buffer.readLine())!= null))
{
   //在每一行后面加上一个"\n"来换行
   resultData += inputLine + "\n";
 }
 //关闭 InputStreamReader
  in.close();
  //关闭 http 连接
  urlConn.disconnect();
```

如果需要使用 POST 方式，则需要 setRequestMethod 设置。代码如下：

```
String httpUrl = "http://192.168.1.110:8080/httpget.jsp";
//获得的数据
String resultData = "";
    URL url = null;
        try  {
            //构造一个 URL 对象
            url = new URL(httpUrl);
            }
                catch (MalformedURLException e) {
                    Log.e(DEBUG_TAG, "MalformedURLException");
                    }
                        if (url!= null) {
                            try  {
                                //使用 HttpURLConnection 打开连接
        HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
                                //因为这个是 post 请求,设立需要设置为 true
                                urlConn.setDoOutput(true);
                                urlConn.setDoInput(true);
                                //设置以 POST 方式
                                urlConn.setRequestMethod("POST");
                                //Post 请求不能使用缓存
                                urlConn.setUseCaches(false);
                                urlConn.setInstanceFollowRedirects(true);
//配置本次连接的 Content - type,配置为 application/x - www - form - urlencoded urlConn
.setRequestProperty("Content - Type","application/x - www - form - urlencoded");
        //连接,从 postUrl.openConnection()至此的配置必须要在连接之前完成
        //要注意的是,connection.getOutputStream 会隐含的进行连接
```

```
                        urlConn.connect();
                        //DataOutputStream 流
            DataOutputStream out = new DataOutputStream(urlConn.getOutputStream());
                        //要上传的参数
                          String content = "par = " + URLEncoder.encode("ABCDEFG",
"gb2312");
                        //将要上传的内容写入流中
                        out.writeBytes(content);
                        //刷新、关闭
                        out.flush();
                        out.close();
                }
        }
```

4. HttpClient 接口

使用 Apache 提供的 HttpClient 接口同样可以进行 HTTP 操作。对于 GET 和 POST 请求方法的操作有所不同。GET 方法的操作代码示例如下：

```
//http 地址
String httpUrl = "http://192.168.1.110:8080/httpget.jsp?par = HttpClient_android_Get";
//HttpGet 连接对象
HttpGet httpRequest = new HttpGet(httpUrl);
//取得 HttpClient 对象
HttpClient httpclient = new DefaultHttpClient();
//请求 HttpClient,取得 HttpResponse
HttpResponse httpResponse = httpclient.execute(httpRequest);
//请求成功
if (httpResponse.getStatusLine().getStatusCode() == HttpStatus.SC_OK){
    //取得返回的字符串
    String strResult = EntityUtils.toString(httpResponse.getEntity());
    mTextView.setText(strResult);
}
else {
    mTextView.setText("请求错误!");
    }
}
```

使用 POST 方法进行参数传递时，需要使用 NameValuePair 来保存要传递的参数。另外，还需要设置所使用的字符集，其代码如下：

```
//http 地址
String httpUrl = "http://192.168.1.110:8080/httpget.jsp";
//HttpPost 连接对象
HttpPost httpRequest = new HttpPost(httpUrl);
//使用 NameValuePair 来保存要传递的 Post 参数
List<NameValuePair> params = new ArrayList<NameValuePair>();
//添加要传递的参数
params.add(new BasicNameValuePair("par", "HttpClient_android_Post"));
//设置字符集
HttpEntity httpentity = new UrlEncodedFormEntity(params, "gb2312");
//请求 httpRequest
httpRequest.setEntity(httpentity);
//取得默认的 HttpClient
HttpClient httpclient = new DefaultHttpClient();
//取得 HttpResponse
```

```
HttpResponse httpResponse = httpclient.execute(httpRequest);
//HttpStatus.SC_OK 表示连接成功
if (httpResponse.getStatusLine().getStatusCode() == HttpStatus.SC_OK)
{
    //取得返回的字符串
    String strResult = EntityUtils.toString(httpResponse.getEntity());
    mTextView.setText(strResult);
}
  else {   mTextView.setText("请求错误!");
  }
}
```

HttpClient 实际上是对 Java 提供方法的一些封装，在 HttpURLConnection 中的输入输出流操作，在这个接口中被统一封装成了 HttpPost(HttpGet)和 HttpResponse，这样，就减少了操作的繁琐性。

另外，在使用 POST 方式进行传输时，需要进行字符编码。

9.4.2 HTTP 通信经典案例

下面通过一个经典示例来演示 GET 请求方式与 POST 请求方式的实现。

【例 9-4】 使用这种方式来查询获取手机号码的基本信息。其具体实现步骤为：

(1) 在 Eclipse 下创建一个 Android 应用项目，命名为 HTTP_GET。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中实现两个 LinearLayout 布局，并声明一个 EditText 控件、一个 ScrollView 控件、一个 TextView 控件和三个 Button 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical" android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/kp" >
    <LinearLayout android:layout_width = "match_parent"
        android:layout_height = "wrap_content">
        <EditText android:id = "@ + id/editText"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
             android:layout_weight = "1"
            android:hint = "输入所需查询的电话号码"
            android:inputType = "text">
            <requestFocus></requestFocus>
        </EditText>
        <Button android:layout_width = "match_parent"
            android:text = "Search"
            android:layout_height = "wrap_content"
            android:id = "@ + id/goQuery"
            android:layout_weight = "3"/>
    </LinearLayout>
    <Button android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:id = "@ + id/Button01"
        android:text = "使用 GET 方式获取信息"/>
    <Button android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
```

```
            android:id = "@ + id/Button02"
            android:text = "使用 POST 方式获取信息"/>
        < ScrollView android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:id = "@ + id/ScrollView1"
            android:scrollbars = "vertical">
            < TextView android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:id = "@ + id/TextView01"/>
        </ScrollView >
</LinearLayout >
```

(3) 打开 src\fs.http_get 包下的 MainActivity.java 文件，在文件中实现 GET 获取信息，其代码为：

```
public class MainActivity extends Activity {
    Button btn_get, btn_post, btn_search;
    EditText edt_input;
    TextView tv_result;
    //查询手机号码信息的基本网址，用于与需要查询的手机号码的拼接
    final static String phoneUrl = "http://api.showji.com/Locating/default.aspx";
    /** 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btn_get = (Button) findViewById(R.id.Button01);
        btn_post = (Button) findViewById(R.id.Button02);
        btn_search = (Button) findViewById(R.id.goQuery);
        edt_input = (EditText) findViewById(R.id.editText);
        tv_result = (TextView) findViewById(R.id.TextView01);
        btn_get.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                //TODO 自动存根法
                Get_url();
            }
        });
        btn_post.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                //TODO 自动存根法
                Post_url();
            }
        });
    }
  //使用 GET 连接查询
    private void Get_url() {
        try {
            //URL
            String phonenum = edt_input.getText().toString();
            phonenum = phonenum.replace(" ", "%20");
            URL geturl = new URL(phoneUrl + "?m = " + phonenum + "&output = xml");
/* 根据拼凑的 URL，打开连接，URL.openConnection 函数会根据 URL 的类型，返回不同的 URLConnection
子类的对象，这里 URL 是一个 http，因此实际返回的是 HttpURLConnection */
            HttpURLConnection httpconn = (HttpURLConnection) geturl
                    .openConnection();
            httpconn.setReadTimeout(10000); //设置超时时间
```

```
            if (httpconn.getResponseCode() == HttpURLConnection.HTTP_OK) {
                Toast.makeText(getApplicationContext(),
                        "GET 连接 手机在线 API 成功!", 1000).show();
                //InputStreamReader,得到数据流
                InputStreamReader isr = new InputStreamReader(httpconn
                        .getInputStream(), "utf-8");
                int i;
                String content = "";
                //read
                while ((i = isr.read())!= -1) {
                    content = content + (char) i;
                }
                isr.close();

                tv_result.setText(content);
            }
            //关闭连接
            httpconn.disconnect();
        } catch (Exception e) {
            Toast.makeText(getApplicationContext(), "GET 连接 手机在线 API 失败",
                    1000).show();
            e.printStackTrace();
        }
    }
```

(4) 在实现网络请求前,必须在 AndroidManifest.xml 文件中申请权限,其代码为:

```
⋮
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name = "android.permission.INTERNET"/>
</manifest>
```

(5) 运行程序,效果如图 9-7 所示。



图 9-7　HTTP 通信

使用 POST 方式来查询获取手机号码的基本信息。其实现步骤为：

打开 src\fs.http_get 包下的 MainActivity.java 文件，添加以下代码：

```
//POST 方式
//查询手机号码信息的 URL 地址
final static String phoneUrl = "http://api.showji.com/Locating/default.aspx";
    private void Post_url() {
        String phonenum = edt_input.getText().toString();
        //构造 URL,直接使用查询信息的网址
        try {
            //构造一个 URL 对象
            URL url = new URL(phoneUrl);
            //使用 HttpURLConnection 打开连接
            HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
            //因为这个是 post 请求,设立需要设置为 true
            urlConn.setDoOutput(true);
            urlConn.setDoInput(true);
            //设置超时时间
            urlConn.setReadTimeout(10000);
            //设置以 POST 方式
            urlConn.setRequestMethod("POST");
            //Post 请求不使用缓存
            urlConn.setUseCaches(false);
            urlConn.setInstanceFollowRedirects(true);
            //配置本次连接的 Content - type,配置为 application/x - www - form - urlencoded
            urlConn.setRequestProperty("Content - Type",
                    "application/x - www - form - urlencoded");
        /* 连接,从 postUrl.openConnection()至此的配置必须要在连接之前完成,要注意的是
connection.getOutputStream 会隐含的进行连接 */
            urlConn.connect();
            //DataOutputStream 流
            DataOutputStream out = new DataOutputStream(urlConn.getOutputStream());
            //要上传的参数,get 方式?之后的内容.m = " + phonenum + "&output = xml"String content
= "m = " + URLEncoder.encode(phonenum, "utf - 8") + "&output = xml";
            //将要上传的内容写入流中
            out.writeBytes(content);
            //刷新、关闭
            out.flush();
            out.close();
            InputStreamReader isr = new InputStreamReader(urlConn
                    .getInputStream());
            int i;
            String content_post = "";
            //read
            while ((i = isr.read())!= - 1) {
                content_post = content_post + (char) i;
            }
            isr.close();
            //设置 TextView
            tv_result.setText(content_post);
            //关闭 http 连接
            urlConn.disconnect();
            Toast.makeText(getApplicationContext(), "POST 连接手机在线 API 成功",
                    1000).show();
        } catch (Exception e) {
```

```
                Toast.makeText(getApplicationContext(), "POST 连接手机在线 API 失败",
                        1000).show();
                e.printStackTrace();
            }
        }
    }
```

下面利用 Android 实现一个在线翻译以及理解战例。

【例 9-5】 当遇到不认识的单词或不理解的词语时，一般会借助于网络来进行查询。具体的实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 li9_5query。

(2) 打开 res\layout 目录下的 main. xml 文件，在文件中声明两个 TextView 控件、一个 EditText 控件、两个 RadioGroup 控件、一个 WebView 控件及一个 Button 控件，其代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "@drawable/kp">
 <TextView
        android:id = "@ + id/textView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_centerHorizontal = "true"
        android:text = "在线查询" />
    <EditText
        android:id = "@ + id/tinput"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_below = "@ + id/textView1"
        android:ems = "10"
        android:hint = "输入要查询的词" />
    <RadioGroup
        android:id = "@ + id/myRadioGroup"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_alignParentRight = "true"
        android:layout_below = "@ + id/tinput"
        android:orientation = "horizontal" >
        <RadioButton
            android:id = "@ + id/myRadioButton1"
            android:layout_height = "wrap_content"
            android:text = "翻译" />
        <RadioButton
            android:id = "@ + id/myRadioButton2"
            android:layout_height = "wrap_content"
            android:text = "百科" />
```

```
        </RadioGroup>
        <Button
            android:id="@+id/submit"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_below="@+id/myRadioGroup"
            android:text="查询  " />
        <TextView
            android:id="@+id/tips"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_below="@+id/submit"
            android:text="查询结果: "
            android:textSize="14sp"
            android:visibility="invisible"
            android:typeface="sans" />
        <WebView
            android:id="@+id/toutput"
            android:layout_width="fill_parent"
            android:layout_height="270px"
            android:layout_alignParentBottom="true"
            android:layout_alignParentLeft="true"
            android:layout_below="@+id/tips"
            android:visibility="invisible" />
    </RelativeLayout>
```

(3) 打开 src\fs.li9_5query 包下的 MainActivity.java 文件，在文件中实现单词的查询与理解，其代码为：

```
package fs.li9_5query;
import android.os.Bundle;
import android.os.Handler;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;
public class MainActivity extends Activity {
    private TextView tips;
    private EditText editText;
    private WebView webView;
    private Button submit;
    RadioButton rb1, rb2;
    RadioGroup rGroup;
    private Handler tHandler = new Handler();
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        webView = (WebView) findViewById(R.id.toutput);
        submit = (Button) findViewById(R.id.submit);
        editText = (EditText) findViewById(R.id.tinput);
        tips = (TextView) findViewById(R.id.tips);
        rb1 = (RadioButton) findViewById(R.id.myRadioButton1);
        rb2 = (RadioButton) findViewById(R.id.myRadioButton2);
        rGroup = (RadioGroup) findViewById(R.id.myRadioGroup);
        rGroup.check(R.id.myRadioButton1);
        WebSettings webSettings = webView.getSettings();
        webSettings.setSaveFormData(false);
        webSettings.setSavePassword(false);
        webSettings.setSupportZoom(false);
        webView.setWebViewClient(new WebViewClient() {
            @Override
            public boolean shouldOverrideUrlLoading(
                    WebView view, String url) {
                //使用自己的 webview 加载
                view.loadUrl(url);
                return true;
            }
        });
        submit.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                if (editText.getText().toString().equals("")) {
                    Toast.makeText(MainActivity.this, "请输入查询的词",
                            Toast.LENGTH_LONG);
                    return;
                }
                tips.setVisibility(TextView.VISIBLE);
                webView.setVisibility(WebView.VISIBLE);
                tHandler.post(new Runnable() {
                    public void run() {
                if (rGroup.getCheckedRadioButtonId() == R.id.myRadioButton1) {
                            webView.loadUrl("http://3g.dict.cn/s.php?q="
                                    + editText.getText().toString());
                        } else {
                            webView.loadUrl("http://www.baike.com/wiki/"
                                    + editText.getText().toString());
                        }
                    }
                });
            }
        });
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //增加了项目操作栏
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

运行程序，效果如图 9-8 所示。



图 9-8　查询界面

第 10 章 Android 开发

智能手机作为目前 Android 系统的最大载体，Android 系统不仅需要满足用户的娱乐需求，更重要也是必须解决手机最基本的两大需求——语音通话和信息，此外还要实现手机的附加功能。

10.1 手机的附加功能

手机的附加功能主要包括查看电池使用情况、手机相关信息、SIM 卡相关信息、计算器、手电筒等。

10.1.1 获取手机信息

每一部手机都有其详细的信息，这些信息包括了手机号码、电信网络等国别等。

TelephonyManager 是一个管理手机通话状态、电话网络信息的服务类。获取 TelephonyManager 十分简单，主要通过 getSystemService 获取 TelephonyManager 对象，接着通过 TelephonyManager 的方法来获取和电信有关的网络信息；然后通过 Android.provider.Setting.System.getString()获取手机的相关设置信息；最后将 setListAdapter 内的信息显示在 ListView 中。

下面通过案例实现：在界面上有一个按钮，当单击按钮时，即可获取手机上的相关信息，信息包括手机号码、电信网络国别、电信公司名称、手机 SIM 码、手机通信类型、手机网络类型、是否漫游以及蓝牙和 WiFi 的状态等。当单击其中的一条信息时，会有 Toast 弹出，给出相关的信息。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Tele_Phone。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中定义一个 RelativeLayout 布局，声明一个 Button 控件及一个 LinearLayout 布局，在 LinearLayout 布局中声明一个 ListView，用于显示手机的相关信息，其代码为：

```
<RelativeLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:background = "#ffcc66">
  <Button
        android:text = "获取手机信息"
        android:id = "@ + id/Button1"
        android:layout_width = "fill_parent"
```

```
            android:layout_height = "wrap_content"/>
        < LinearLayout
            android:orientation = "vertical"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:background = " # ffcc66"
            android:paddingLeft = "0dip"
            android:paddingRight = "5dip"
            android:paddingTop = "5dip">
        < ListView
           android:id = "@ + id/ListView1"
           android:layout_width = "wrap_content"
           android:layout_height = "wrap_content"
           android:cacheColorHint = " # ffcc66"/>
        </LinearLayout >
    </RelativeLayout >
```

(3) 打开 src\fs. tele_phone 包下的 MainActivity. java 文件，在文件中实现获取手机的相关信息将其显示在 ListView 控件中，其代码为：

```
package fs. tele_phone;
import java. util. ArrayList;
import java. util. List;
import android. app. Activity;
import android. content. ContentResolver;
import android. graphics. Color;
import android. os. Bundle;
import android. telephony. TelephonyManager;
import android. view. Gravity;
import android. view. View;
import android. view. ViewGroup;
import android. view. View. OnClickListener;
import android. widget. AdapterView;
import android. widget. BaseAdapter;
import android. widget. Button;
import android. widget. LinearLayout;
import android. widget. ListView;
import android. widget. TextView;
import android. widget. Toast;
import android. widget. AdapterView. OnItemClickListener;
public class MainActivity extends Activity {
    private ListView lv;
    private TelephonyManager tm;
    private ContentResolver cr;
    private List < String > list = new ArrayList < String >();
    private List < String > name = new ArrayList < String >();
    private Button bCheck;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super. onCreate(savedInstanceState);
        setContentView(R. layout. main);
        lv = (ListView)this. findViewById(R. id. ListView1);
        tm = (TelephonyManager)getSystemService(TELEPHONY_SERVICE);
        cr = MainActivity. this. getContentResolver();
        bCheck = (Button)this. findViewById(R. id. Button1);
        String str = null;                                //记录前面定义的变量
```

```
        name.add("手机号码：");
        name.add("电信网络国别：");
        name.add("电信公司代码：");
        name.add("电信公司名称：");
        name.add("SIM 码：");
        name.add("手机通信类型：");
        name.add("手机网络类型 ：");
        name.add("手机是否漫游：");
        name.add("蓝牙状态：");
        name.add("WIFI 状态：");
        if(tm.getLine1Number()!= null)                    //手机号码
        {
            list.add(tm.getLine1Number());
        }else
        {
            list.add("无法取得您的电话号码");
        }
        if(!tm.getNetworkCountryIso().equals(""))         //电信网络国别
        {
            list.add(tm.getNetworkCountryIso());
        }else
        {
            list.add("无法取得您的电信网络国别");
        }
        if(!tm.getNetworkOperator().equals(""))           //电信公司代码
        {
            list.add(tm.getNetworkOperator());
        }else
        {
            list.add("无法获取电信公司代码");
        }
        if(!tm.getNetworkOperatorName().equals(""))       //电信公司名称
        {
            list.add(tm.getNetworkOperatorName());
        }else
        {
            list.add("无法获取电信公司名称");
        }
        if(tm.getSimSerialNumber()!= null)                //手机 SIM 码
        {
            list.add(tm.getSimSerialNumber());
        }else
        {
            list.add("无法获取手机 SIM 码");
        }
    if(tm.getPhoneType() == TelephonyManager.PHONE_TYPE_GSM)  //手机行动通信类型
        {
            list.add("GSM");
        }
        else
        {
            list.add("无法获取手机通信类型");
        }
      //获取手机网络类型
        if(tm.getNetworkType() == TelephonyManager.NETWORK_TYPE_EDGE)
```

```
        {
            list.add("EDGE");
        }else if(tm.getNetworkType() == TelephonyManager.NETWORK_TYPE_GPRS)
        {
            list.add("GPRS");
        }else if(tm.getNetworkType() == TelephonyManager.NETWORK_TYPE_UMTS)
        {
            list.add("UMTS");
        }
        else
        {
            list.add("无法获取手机网络类型");
        }

        if(tm.isNetworkRoaming())                          //手机是否漫游
        {
            list.add("手机漫游中");
        }else
        {
            list.add("手机无漫游");
        }
        str = android.provider.Settings.System.getString(
                cr,android.provider.Settings.System.BLUETOOTH_ON
                );
        if(str.equals("1"))
        {
            list.add("蓝牙已打开");
        }else
        {
            list.add("蓝牙未打开");
        }
        str = android.provider.Settings.System.getString(cr, android.provider.Settings.
System.WIFI_ON);
        if(str.equals("1"))
        {
            list.add("WIFI 已打开");
        }else
        {
            list.add("WIFI 未打开");
        }
        bCheck.setOnClickListener
        (
                new OnClickListener()
                {
                    public void onClick(View v) {
                        BaseAdapter ba = new BaseAdapter()         //创建适配器
                        {
                            public int getCount() {
                                return list.size();
                            }
                            public Object getItem(int position) {
                                return null;
                            }
                            public long getItemId(int position) {
                                return 0;
```

```
                    }
                public View getView(int arg0, View arg1, ViewGroup arg2) {
                    LinearLayout ll = new LinearLayout(MainActivity.this);
                        ll.setOrientation(LinearLayout.HORIZONTAL);
                        ll.setPadding(5, 5, 5, 5);
        TextView tv = new TextView(MainActivity.this);          //初始化 TextView
                        tv.setTextColor(Color.BLACK);   //设置字体颜色
                        tv.setPadding(5,5,5,5);
                        tv.setText(name.get(arg0));     //添加任务名字
                        tv.setGravity(Gravity.LEFT);    //左对齐
                        tv.setTextSize(18);             //字体大小
                        ll.addView(tv);                 //LinearLayout 添加 TextView
        TextView tvv = new TextView(MainActivity.this);         //初始化 TextView
                        tvv.setTextColor(Color.BLACK);  //设置字体颜色
                        tvv.setPadding(5,5,5,5);
                        tvv.setText(list.get(arg0));    //添加任务名字
                        tvv.setGravity(Gravity.LEFT);   //左对齐
                        tvv.setTextSize(18);            //字体大小
                        ll.addView(tvv);                //LinearLayout 添加 TextView
                        return ll;
                    }
                };
                lv.setAdapter(ba);                      //设置适配器
                lv.setOnItemClickListener               //设置选中菜单的监听器
                (
                        new OnItemClickListener()
                        {
                public void onItemClick(AdapterView<?> arg0, View arg1,
                                        int arg2, long arg3) {
              Toast.makeText(MainActivity.this, name.get(arg2) + "" + list.get(arg2),
Toast.LENGTH_SHORT).show();
                            }
                        }
                );
            }
        }
    );
  }
}
```

(4) 打开 AndroidManifest.xml 文件，为程序设置权限。添加代码为：

```
⋮
        </activity>
    </application>
<!-- 添加权限 -->
<uses-permission android:name="android.permission.READ_PHONE_STATE"></uses-permission>
</manifest>
```

运行程序，效果如图 10-1(a)所示。单击界面中的“查看手机信息”按钮，效果如图 10-1(b)所示；单击任何一项信息，即显示对应的 Toast 提示，效果如图 10-1(c)所示。

10.1.2 读取 SIM 卡参数

SIM(Subscriber Identity Module，客户识别模块)卡也称为用户身份识别卡、智能卡，GSM 数字移动电话机必须装上此卡方能使用。它在一芯片上存储了数字移动电话客户的信



(a) 默认界面　　(b) 相关信息显示　　(c) Toast提示

图 10-1　显示手机相关信息

息、加密的密钥以及用户的电话簿等内容，可供 GSM 网络客户身份进行鉴别，并对客户通话时的语音信息进行加密。

手机的 SIM 卡是手机的一个重要组成部分，没有 SIM 卡不能正常拨打电话。本节将介绍获取 SIM 卡的信息的方法。

该案例通过使用 TelephonyManager 获取手机 SIM 卡的相关信息，并将获得的 SIM 卡的状态、卡号、SIM 卡供应商、SIM 卡供应商名称、SIM 卡国别以 ListView 形式呈现在界面上。使用 TelephonyManager 获取手机 SIM 卡的信息需要为手机添加权限声明，权限代码为 <uses-permission android:name="android.permission.READ_PHONE_STATE"/>。

读取 SIM 卡参数的具体步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 SIM_Message。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中定义一个 RelativeLayout 布局，在声明一个 Button 控件及一个 LinearLayout 布局，在 LinearLayout 布局中声明一个 ListView，用于显示手机的相关信息，其代码为：

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffcc66">
  <Button
        android:text="获取 SIM 卡信息"
        android:id="@+id/Button1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"/>
  <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
```

```
            android:layout_height = "wrap_content"
            android:background = "#ffcc66"
            android:paddingLeft = "0dip"
            android:paddingRight = "5dip"
            android:paddingTop = "5dip">
        <ListView
            android:id = "@+id/ListView1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:cacheColorHint = "#ffcc66"/>
    </LinearLayout>
</RelativeLayout>
```

（3）打开 src\fs. sim_message 包下的 MainActivity. java 文件，在文件中实现获取 SIM 卡的相关信息将其显示在 ListView 控件中，其代码为：

```
package fs.sim_message;
import java.util.ArrayList;
import java.util.List;
import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.telephony.TelephonyManager;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;
public class MainActivity extends Activity {
    private ListView lv;
    private TelephonyManager tm;
    private List<String> list = new ArrayList<String>();
    private List<String> name = new ArrayList<String>();
    private Button bCheck;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        lv = (ListView)this.findViewById(R.id.ListView1);
        tm = (TelephonyManager)getSystemService(TELEPHONY_SERVICE);
        bCheck = (Button)this.findViewById(R.id.Button1);
        name.add("SIM 卡的状态：");
        name.add("SIM 卡号：");
        name.add("SIM 卡供应商号：");
        name.add("SIM 卡供应商名称：");
        name.add("SIM 卡国别：");
        /* 通过 TelephonyManager 对象判断 SIM 卡状态、SIM 卡卡号、SIM 卡供应商代号、供应商名称
以及 SIM 卡国别，最后获取的信息添加到 list 列表 */
        if(tm.getSimState() == TelephonyManager.SIM_STATE_READY)  //SIM 卡状态
```

```
{
    list.add("状态良好");
}else if(tm.getSimState() == TelephonyManager.SIM_STATE_ABSENT)
{
    list.add("您目前没有 SIM 卡");
}else if(tm.getSimState() == TelephonyManager.SIM_STATE_UNKNOWN)
{
    list.add("SIM 卡处于未知状态");
}
if(tm.getSimSerialNumber()!= null)                     //SIM 卡卡号
{
    list.add(tm.getSimSerialNumber());
}else
{
    list.add("没有 SIM 卡卡号");
}
if(!tm.getSimOperator().equals(""))                    //SIM 卡供应商代号
{
    list.add(tm.getSimOperator());
}else
{
    list.add("没有 SIM 卡供应商代号");
}
if(!tm.getSimOperatorName().equals(""))                //SIM 卡供应商名称
{
    list.add(tm.getSimOperatorName());
}else
{
    list.add("没有 SIM 卡供应商名称");
}
if(!tm.getSimCountryIso().equals(""))
{
    list.add(tm.getSimCountryIso());
}else
{
    list.add("无法获取 SIM 国别");
}
 bCheck.setOnClickListener
(
        new OnClickListener()
        {
            public void onClick(View v) {
                BaseAdapter ba = new BaseAdapter()       //创建适配器
                {
                    public int getCount() {
                        return list.size();
                    }
                    public Object getItem(int position) {
                        return null;
                    }
                    public long getItemId(int position) {
                        return 0;
                    }
                public View getView(int arg0, View arg1, ViewGroup arg2) {
                    LinearLayout ll = new LinearLayout(MainActivity.this);
```

```
                    ll.setOrientation(LinearLayout.HORIZONTAL);
                    ll.setPadding(5, 5, 5, 5);
        TextView tv = new TextView(MainActivity.this);    //初始化 TextView
                    tv.setTextColor(Color.BLACK);    //设置字体颜色
                    tv.setPadding(5,5,5,5);
                    tv.setText(name.get(arg0));     //添加任务名字
                    tv.setGravity(Gravity.LEFT);    //左对齐
                    tv.setTextSize(18);             //字体大小
                    ll.addView(tv);                 //LinearLayout 添加 TextView
          TextView tvv = new TextView(MainActivity.this);   //初始化 TextView
                    tvv.setTextColor(Color.BLACK);      //设置字体颜色
                    tvv.setPadding(5,5,5,5);
                    tvv.setText(list.get(arg0));        //添加任务名字
                    tvv.setGravity(Gravity.LEFT);       //左对齐
                    tvv.setTextSize(18);                //字体大小
                    ll.addView(tvv);        //LinearLayout 添加 TextView
                    return ll;
                }
            };
            lv.setAdapter(ba);              //设置适配器
            lv.setOnItemClickListener       //设置选中菜单的监听器
            (
                    new OnItemClickListener()
                    {
            public void onItemClick(AdapterView<?> arg0, View arg1,
                                int arg2, long arg3) {
                              Toast.makeText(MainActivity.this, name.get(arg2)
+ "" + list.get(arg2), Toast.LENGTH_SHORT).show();
                         }
                    }
            );
        }

      }
    );
  }
}
```

(4) 打开 AndroidManifest.xml 文件,设置 SIM 卡权限。添加代码为:

```
⋮
   </application>
<!-- 添加权限 -->
<uses-permission android:name = "android.permission.READ_PHONE_STATE"/>
</manifest>
```

运行程序,效果如图 10-2(a)所示。当单击界面中的"获取 SIM 卡信息"按钮时,效果如图 10-2(b)所示;当单击任何一项时即弹出相应的 Toast 说明,如图 10-2(c)所示。

10.1.3 查看电池

手机在使用过程中,最担心的是没电而影响业务,所以及时显示电池容量功能是非常必要的。

可以使用 Android API 中的 BroadcastReseiver 类和 Button 的 Listener 类,当 Reseiver 被注册后会在后台等待被其他程序调用。当指定要捕捉的 Action 发生时,Reseiver 就会被调



(a) 默认界面　(b) SIM卡信息　(c) Toast提示

图 10-2　显示 SIM 相关信息

用，并运行 onReseiver 来实现里面的程序。

在实例中，将利用 BroadcastReseiver 的特性来获取手机电池的容量。通过注册 BroadcastReseiver 时设置的 IntentFiler 来获取系统发出的 Intent. ACTION_BATTERY_CHANGED，然后以此来获取电池的容量、温度及电压等。其具体操作步骤为：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 Souces_Message。

（2）打开 res\layout 目录下的 main. xml 布局文件，在文件声明一个 TextView 控件。用于显示电池情况，其代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#ffcc66">
    <TextView
        android:id = "@ + id/TV"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
</RelativeLayout>
```

（3）打开 src\fs. souces_message 包下的 MainActivity. java 文件，在文件中实现显示电池的电量、温度及电压，其代码为：

```
package fs.souces_message;
import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
```

```java
import android.content.Intent;
import android.content.IntentFilter;
import android.os.BatteryManager;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends Activity
{
    private int BatteryN;                          //目前电量
    private int BatteryV;                          //电池电压
    private double BatteryT;                       //电池温度
    private String BatteryStatus;                  //电池状态
    private String BatteryTemp;                    //电池使用情况
    public TextView TV;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        /* 注册一个系统 BroadcastReceiver,作为访问电池计量之用这个不能直接在 AndroidManifest.
xml 中注册 */
        registerReceiver(mBatInfoReceiver, new IntentFilter(Intent.ACTION_BATTERY_CHANGED));
        TV = (TextView)findViewById(R.id.TV);
    }
    /* 创建广播接收器 */
    private BroadcastReceiver mBatInfoReceiver = new BroadcastReceiver()
    {
        public void onReceive(Context context, Intent intent)
        {
            String action = intent.getAction();
            /*
             * 如果捕捉到的 action 是 ACTION_BATTERY_CHANGED, 就运行 onBatteryInfoReceiver()
             */
            if (Intent.ACTION_BATTERY_CHANGED.equals(action))
            {
                BatteryN = intent.getIntExtra("level", 0);        //目前电量
                BatteryV = intent.getIntExtra("voltage", 0);      //电池电压
                BatteryT = intent.getIntExtra("temperature", 0);  //电池温度
                switch (intent.getIntExtra("status", BatteryManager.BATTERY_STATUS_UNKNOWN))
                {
                case BatteryManager.BATTERY_STATUS_CHARGING:
                    BatteryStatus = "充电状态";
                    break;
                case BatteryManager.BATTERY_STATUS_DISCHARGING:
                    BatteryStatus = "放电状态";
                    break;
                case BatteryManager.BATTERY_STATUS_NOT_CHARGING:
                    BatteryStatus = "未充电";
                    break;
                case BatteryManager.BATTERY_STATUS_FULL:
                    BatteryStatus = "充满电";
                    break;
                case BatteryManager.BATTERY_STATUS_UNKNOWN:
                    BatteryStatus = "未知状态";
                    break;
                }
```

```
                    switch (intent.getIntExtra("health", BatteryManager.BATTERY_HEALTH_UNKNOWN))
                    {
                    case BatteryManager.BATTERY_HEALTH_UNKNOWN:
                        BatteryTemp = "未知错误";
                        break;
                    case BatteryManager.BATTERY_HEALTH_GOOD:
                        BatteryTemp = "状态良好";
                        break;
                    case BatteryManager.BATTERY_HEALTH_DEAD:
                        BatteryTemp = "电池没有电";
                        break;
                    case BatteryManager.BATTERY_HEALTH_OVER_VOLTAGE:
                        BatteryTemp = "电池电压过高";
                        break;
                    case BatteryManager.BATTERY_HEALTH_OVERHEAT:
                        BatteryTemp = "电池过热";
                        break;
                    }
            TV.setText("目前电量为" + BatteryN + "% --- " + BatteryStatus + "\n" + "电压为" +
BatteryV + "mV --- " + BatteryTemp + "\n" + "温度为" + (BatteryT * 0.1) + "℃");
                }
            }
        };
}
```

运行程序，效果如图10-3所示。



图10-3　电池情况

10.1.4　开机启动服务

目前，需要开发一个开机自启动的GTD应用程序来提醒用户重要的日程安排，对于这类应用，Android提供了一个BroadcastReceiver组件来对应用程序的运行环境进行监听，并对各种事件进行对应的处理。使用BroadcastReceiver十分简单，只需要在AndroidManifest.xml或代码中进行相应的注册(这也是Android开发的两种方式)。然后，在广播事件到来时，就能通过重写BroadcastReceiver的onReceive()方法来执行相应的操作。

下面简单来演示如何开发开机自启动应用。其具体操作步骤为：

(1) 在Eclipse中创建一个Android应用项目，命名为Boot_Test。

(2) 打开res\layout目录下的main.xml布局文件，在文件中声明一个TextView控件，用于提示内容，其代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
```

```
        android:paddingTop = "@dimen/activity_vertical_margin"
        tools:context = ".MainActivity"
        android:background = "#ffcc66">
    <TextView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:id = "@ + id/mytv"/>
</RelativeLayout>
```

(3) 打开 src\fs. boot_test 包下的 MainActivity. java 文件，在文件中实现获取 TextView 对象内容，其代码为：

```
package fs.com.boot_test;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends Activity
{
    private TextView tv;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello. I started!");
        setContentView(tv);
       //加载 main.xml 布局文件
        setContentView(R.layout.main);
        //通过 findViewById()方法得到 TextView 对象
       tv = (TextView)findViewById(R.id.mytv);
       //设置 TextView 对象的文本
       tv.setText(R.string.hello_world);
       }
}
```

(4) 在 src\fs. boot_test 包下的创建一个接收广播类文件，命名为 Broadcast_Receiver. java 文件，其代码为：

```
package fs.com.boot_test;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
public class Broadcast_Receiver extends BroadcastReceiver
{
    static final String ACTION = "android.intent.action.BOOT_COMPLETED";
    @Override
    public void onReceive(Context context, Intent intent)
    {
        if (intent.getAction().equals(ACTION))
        {
            //创建一个 Intent 对象,并指定要启动的 class
            Intent myintent = new Intent(context,MainActivity.class);
            //调用另外一个 Activity,将主控权移交给 MainActivity
            context.startActivity(myintent);
            }
        }
    }
```

(5) 打开项目配置 AndroidManifest.xml 文件,设置权限,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.com.boot_test"
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    < application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        < activity
            android:name = "fs.com.boot_test.MainActivity"
            android:label = "@string/app_name" >
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" />
                < action android:name = "android.intent.action.BOOT_COMPLETED" />
                < category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
        </activity >
    </application >
    < uses - permission android:name = "android.permission.RECEIVE_BOOT_COMPLETED"/>
</manifest >
```

至此,在第二次打开 Android 的模拟器时,即可弹出提示内容。

10.1.5 闹钟设置

闹钟在生活中最常见了,在 Android 中可以通过 AlarmManager 实现闹钟,AlarmManager 类用来设置在某个指定的时间内实现提示功能。AlarmManager 通过 onReceive()方法去执行这些事件,就算系统处于待机状态,同样不会影响运行,可以通过 Context.getSystemService 方法来获得该服务。AlarmManager 中的方法如表 10-1 所示。

表 10-1 设置闹钟方法及说明

方法名称	说明	方法名称	说明
Cancel	取消 AlarmManager 服务	SetRepeating	设置精确周期
Set	设置 AlarmManager 服务	setTimeZone	设置时区
setInexactRepeating	设置不精确周期		

实现闹钟,首先需要创建一个继承自 BroadcastReceiver 的类,实现 onReceive 方法来接受这个 Alarm 服务,然后通过建立 Intent 和 PendingIntent 连接来调用 Alarm 组件。通过 TimerPickerDialog 来设置闹铃时间,当时间到了指定的时间后 onReceiver 方法跳转到 Alarm 服务界面。

下面案例用于实现 Android 闹钟的设置。具体操作步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 Alarm_Clock。

(2) 打开 res\layout 目录下的 main.xml 布局文件,在文件中声明两个 Button 控件,其代码为:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffcc66">
<Button
    android:text="设置闹钟"
    android:id="@+id/Button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
<Button
    android:text="取消闹钟"
    android:id="@+id/Button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
</LinearLayout>
```

（3）打开 src\alarm_clock 包下的 MainActivity.java 文件，在文件中设置闹铃和取消闹铃的实践进行监听，其代码为：

```
package fs.alarm_clock;
import android.app.Activity;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
/**
 * 测试 AlarmManager
 */
public class MainActivity extends Activity {
    //声明 Button
    private Button setBtn, cancelBtn;
    //定义广播 Action
private static final String BC_ACTION = "com.amaker.ch08.app.action.BC_ACTION";
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //设置当前布局视图
        setContentView(R.layout.main);
        //实例化 Button
        setBtn = (Button) findViewById(R.id.Button1);
        cancelBtn = (Button) findViewById(R.id.Button2);
        //获得 AlarmManager 实例
    final AlarmManager am = (AlarmManager) getSystemService(ALARM_SERVICE);
        //实例化 Intent
        Intent intent = new Intent();
        //设置 Intent action 属性
        intent.setAction(BC_ACTION);
        intent.putExtra("msg", "时间到了!");
        //实例化 PendingIntent
    final PendingIntent pi = PendingIntent.getBroadcast(MainActivity.this, 0,intent, 0);
        //获得系统时间
        final long time = System.currentTimeMillis();
        //设置按钮单击事件
```

```
            setBtn.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    //重复提示,从当前时间开始,间隔 5 秒
                    am.setRepeating(AlarmManager.RTC_WAKEUP, time,8 * 1000, pi);
                    }
                });
            //设置按钮单击事件
            cancelBtn.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    am.cancel(pi);
                    }
            });
        }
    }
```

(4) 在 src\alarm_clock 包下的创建一个新类,命名为 ClockR.java,在文件中实现 Toast 消息的提示,其代码为:

```
package fs.alarm_clock;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;
public class ClockR extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        //获得提示信息
        String msg = intent.getStringExtra("msg");
        //显示提示信息
        Toast.makeText(context, msg, Toast.LENGTH_LONG).show();
        }
}
```

(5) 打开 AndroidManifest.xml 项目设置文件,在文件中添加对应的权限,添加代码为:

```
 ⋮
        </activity>
        <receiver android:name = "ClockR">
     <intent - filter>
        <action android:name = "com.amaker.ch08.app.action.BC_ACTION"/>
        </intent - filter>
    </receiver>
</application>
</manifest>
```

运行程序,效果如图 10-4(a)所示。当单击界面中的“设置闹钟”按钮时,效果如图 10-4(b)所示。

10.1.6 万年日历实现

Android 平台上几种有用的日历控件。日历控件在 Web 开发中有多种解决方案,而且容易实现,但是在 Android 平台上的解决方案较少且不容易实现。

在 Android 3.0 中才新增了日历视图控件,可以显示网格状的日历内容,那么对于 Android 3.0 以下的版本要使用日历控件只能借助第三方,目前应用最多的是 CalendarView。



(a) 默认界面



(b) 设置闹钟

图 10-4　闹钟设置效果

先简单介绍下 CalendarView 日历控件的使用。

android. widget. CalendarView 是从 android. widget. FrameLayout 中继承的。CalendarView 类提供了基本的日历设置方法,如表 10-2 所示。

表 10-2　日历设置方法名称及说明

方 法 名 称	说　　明
long getDate()	获取从 1970 年,1 月 1 日,0 点 0 分 0 秒到现在的毫秒数,因为返回是 long 型最终只能截止到 2038 年
int getFirstDayOfWeek()	获取当天是本周的第几天,Android123 提示返回的定义在 java. util. Calendar 类中。例如,Calendar. Monday 为星期一,定义值为 2
long getMaxDate()	获取 CalendarView 支持 1970 年到那天的最大天数
long getMinDate()	获取 CalendarView 支持 1970 年到那天的最小天数
boolean getShowWeekNumber()	获取是否显示星期
boolean isEnabled()	是否显示本日历视图
void setDate(long date, boolean animate, boolean center)	设置选择日期到 1970 年的描述
setDate(long date)	设置选择的日期描述到 1970 年
setEnabled(boolean enabled)	设置是否启用视图
setFirstDayOfWeek(int firstDayOfWeek)	设置本周起始天数
setMaxDate(long maxDate)	设置最大的日期
setMinDate(long minDate)	设置最小的日期
setOnDateChangeListener (CalendarView. OnDateChangeListener listener)	日历视图修改的接口
void setShowWeekNumber(boolean showWeekNumber)	设置是否显示周号

下面通过一个案例来演示在 Android 实现万能日历的设置。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Calendar_Manager。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 TextView 控件用来显示日历界面，以及声明两个 Button 控件用于实现翻页，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:gravity = "center_horizontal"
    android:orientation = "vertical" >
    <RelativeLayout
        android:id = "@ + id/relativeLayout1"
        android:layout_width = "match_parent"
        android:layout_height = "40dip"
        android:background = " # EDE8DD" >
        <TextView
            android:id = "@ + id/Top_Date"
            android:layout_width = "150dip"
            android:layout_height = "wrap_content"
            android:layout_centerInParent = "true"
            android:gravity = "center_horizontal|center"
            android:textColor = " # 424139"
            android:textSize = "19sp"
            android:textStyle = "bold" />
        <Button
            android:id = "@ + id/btn_pre_month"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_alignParentLeft = "true"
            android:layout_centerVertical = "true"
            android:layout_marginLeft = "30dp"
            android:background = "@drawable/bt11" />
        <Button
            android:id = "@ + id/btn_next_month"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_alignParentRight = "true"
            android:layout_centerVertical = "true"
            android:layout_marginRight = "30dp"
            android:background = "@drawable/bt12" />
    </RelativeLayout>
</LinearLayout>
```

(3) 在 res\values 目录下新建一个颜色渐变文件，命名为 colors.xml 文件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <!-- 主应用程序 -->
    <color name = "title_icon"># ff000000 </color>
    <color name = "text_minor"># ff666666 </color>
    <color name = "application_backcolor"># fff6eade </color>
    <color name = "login_font"># 3D3E40 </color>
    <!-- 日历 -->
    <color name = "calendar_background"># 000000 </color>
    <color name = "recordremind_background"># FFFFFF </color>
```

```
    <color name = "border_color">#FFFFFF</color>
    <color name = "weekname_color">#FFFFFF</color>
    <color name = "day_color">#FFFFFF</color>
    <color name = "inner_grid_color">#FFFFFF</color>
    <color name = "prev_next_month_day_color">#999999</color>
    <color name = "text_color">#FFFF00</color>
    <color name = "recordremindtext_color">#000000</color>
    <color name = "current_day_color">#FF0000</color>
    <color name = "today_background_color">#FF0000</color>
    <color name = "today_color">#FFFFFF</color>
    <color name = "sunday_saturday_color">#FF0000</color>
    <color name = "sunday_saturday_prev_next_month_day_color">#990000</color>
    <color name = "transparent">#50000000</color>
    <color name = "white">#ffffff</color>
    <color name = "black">#000000</color>
    <color name = "bgcolor">#ffffff</color>
    <color name = "txtcolor">#000000</color>
    <color name = "red">#ff0000</color>
    <color name = "Calendar_WeekBgColor">#EFE7DE</color>
    <color name = "Calendar_WeekFontColor">#8C8A8C</color>
    <color name = "Calendar_DayBgColor">#F7F7F7</color>
    <color name = "isHoliday_BgColor">#E0E0E0</color>
    <color name = "unPresentMonth_FontColor">#8C8A8C</color>
    <color name = "isPresentMonth_FontColor">#000000</color>
    <color name = "isToday_BgColor">#EBDCC1</color>
    <color name = "specialReminder">#FF0000</color>
    <color name = "commonReminder">#BDBAB5</color>
    <color name = "gray">#808080</color>
</resources>
```

(4) 打开 src\calendar_manager 包下的 MainActivity.java，在文件中实现日历生成、外层容器、当前日期操作、页面控件、数据源等功能，其代码为：

```
⋮
public class MainActivity extends Activity{
    //生成日历,外层容器
    private LinearLayout layContent = null;
    private ArrayList<DateWidgetDayCell> days = new ArrayList<DateWidgetDayCell>();
    //日期变量
    public static Calendar calStartDate = Calendar.getInstance();
    private Calendar calToday = Calendar.getInstance();
    private Calendar calCalendar = Calendar.getInstance();
    private Calendar calSelected = Calendar.getInstance();
    //当前操作日期
    private int iMonthViewCurrentMonth = 0;
    private int iMonthViewCurrentYear = 0;
    private int iFirstDayOfWeek = Calendar.MONDAY;
    private int Calendar_Width = 0;
    private int Cell_Width = 0;
    //页面控件
    TextView Top_Date = null;
    Button btn_pre_month = null;
    Button btn_next_month = null;
    TextView arrange_text = null;
    LinearLayout mainLayout = null;
    LinearLayout arrange_layout = null;
```

```
//数据源
ArrayList<String> Calendar_Source = null;
Hashtable<Integer, Integer> calendar_Hashtable = new Hashtable<Integer, Integer>();
Boolean[] flag = null;
Calendar startDate = null;
Calendar endDate = null;
int dayvalue = -1;
public static int Calendar_WeekBgColor = 0;
public static int Calendar_DayBgColor = 0;
public static int isHoliday_BgColor = 0;
public static int unPresentMonth_FontColor = 0;
public static int isPresentMonth_FontColor = 0;
public static int isToday_BgColor = 0;
public static int special_Reminder = 0;
public static int common_Reminder = 0;
public static int Calendar_WeekFontColor = 0;
String UserName = "";
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //获得屏幕宽和高,并计算屏幕宽度分 7 等份的大小
    WindowManager windowManager = getWindowManager();
    Display display = windowManager.getDefaultDisplay();
    int screenWidth = display.getWidth();
    Calendar_Width = screenWidth;
    Cell_Width = Calendar_Width / 7 + 1;
    //定制布局文件,并设置属性
    mainLayout = (LinearLayout) getLayoutInflater().inflate(R.layout.main, null);
    setContentView(mainLayout);
    //声明控件,并绑定事件
    Top_Date = (TextView) findViewById(R.id.Top_Date);
    btn_pre_month = (Button) findViewById(R.id.btn_pre_month);
    btn_next_month = (Button) findViewById(R.id.btn_next_month);
    btn_pre_month.setOnClickListener(new Pre_MonthOnClickListener());
    btn_next_month.setOnClickListener(new Next_MonthOnClickListener());
    //计算本月日历中的第一天(一般是上月的某天),并更新日历
    calStartDate = getCalendarStartDate();
    mainLayout.addView(generateCalendarMain());
    DateWidgetDayCell daySelected = updateCalendar();
    if (daySelected!= null)
        daySelected.requestFocus();
    LinearLayout.LayoutParams Param1 = new LinearLayout.LayoutParams(
            ViewGroup.LayoutParams.MATCH_PARENT,
            ViewGroup.LayoutParams.MATCH_PARENT);
    ScrollView view = new ScrollView(this);
    arrange_layout = createLayout(LinearLayout.VERTICAL);
    arrange_layout.setPadding(5, 2, 0, 0);
    arrange_text = new TextView(this);
    mainLayout.setBackgroundColor(Color.WHITE);
    arrange_text.setTextColor(Color.BLACK);
    arrange_text.setTextSize(18);
    arrange_layout.addView(arrange_text);
    startDate = GetStartDate();
    calToday = GetTodayDate();
    endDate = GetEndDate(startDate);
```

```
        view.addView(arrange_layout, Param1);
        mainLayout.addView(view);
    ⋮
```

(5) 在 src\calendar_manager 包下创建一个 DateWidgetDayCell.java 文件，该文件用于实现日期控件字串，其代码为：

```
⋮
public class DateWidgetDayCell extends View {
    //字体大小
    private static final int fTextSize = 28;
    //基本元素
    private OnItemClick itemClick = null;
    private Paint pt = new Paint();
    private RectF rect = new RectF();
    private String sDate = "";
    //当前日期
    private int iDateYear = 0;
    private int iDateMonth = 0;
    private int iDateDay = 0;
    //布尔变量
    private boolean bSelected = false;
    private boolean bIsActiveMonth = false;
    private boolean bToday = false;
    private boolean bTouchedDown = false;
    private boolean bHoliday = false;
    private boolean hasRecord = false;
    public static int ANIM_ALPHA_DURATION = 100;
    public interface OnItemClick {
        public void OnClick(DateWidgetDayCell item);
    }
    //构造函数
    public DateWidgetDayCell(Context context, int iWidth, int iHeight) {
        super(context);
        setFocusable(true);
        setLayoutParams(new LayoutParams(iWidth, iHeight));
    }
    //取变量值
    public Calendar getDate() {
        Calendar calDate = Calendar.getInstance();
        calDate.clear();
        calDate.set(Calendar.YEAR, iDateYear);
        calDate.set(Calendar.MONTH, iDateMonth);
        calDate.set(Calendar.DAY_OF_MONTH, iDateDay);
        return calDate;
    }
⋮
```

(6) 在 src\calendar_manager 包下创建一个 DateWidgetDayHeader.java 文件，该文件用于设置日期控件的字串头，其代码为：

```
package fs.calendar_manager;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.RectF;
```

```
import android.view.View;
import android.widget.LinearLayout.LayoutParams;
public class DateWidgetDayHeader extends View {
    //字体大小
    private final static int fTextSize = 22;
    private Paint pt = new Paint();
    private RectF rect = new RectF();
    private int iWeekDay = -1;
    public DateWidgetDayHeader(Context context, int iWidth, int iHeight) {
        super(context);
        setLayoutParams(new LayoutParams(iWidth, iHeight));
    }
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        //设置矩形大小
        rect.set(0, 0, this.getWidth(), this.getHeight());
        rect.inset(1, 1);
        //绘制日历头部
        drawDayHeader(canvas);
    }
    private void drawDayHeader(Canvas canvas) {
        //画矩形,并设置矩形画笔的颜色
        pt.setColor(MainActivity.Calendar_WeekBgColor);
        canvas.drawRect(rect, pt);
        //写入日历头部,设置画笔参数
        pt.setTypeface(null);
        pt.setTextSize(fTextSize);
        pt.setAntiAlias(true);
        pt.setFakeBoldText(true);
        pt.setColor(MainActivity.Calendar_WeekFontColor);
        final String sDayName = DayStyle.getWeekDayName(iWeekDay);
        final int iPosX = (int) rect.left + ((int) rect.width() >> 1)
                - ((int) pt.measureText(sDayName) >> 1);
        final int iPosY = (int) (this.getHeight()
                - (this.getHeight() - getTextHeight())/2 - pt
                .getFontMetrics().bottom);
        canvas.drawText(sDayName, iPosX, iPosY, pt);
    }
    //得到字体高度
    private int getTextHeight() {
        return (int) (-pt.ascent() + pt.descent());
    }
    //得到一星期的第几天的文本标记
    public void setData(int iWeekDay) {
        this.iWeekDay = iWeekDay;
    }
}
```

(7) 在src\calendar_manager包下创建一个DayStyle.java,该文件用于实现日期类型,其代码为:

```
package fs.calendar_manager;
import java.util.Calendar;
public class DayStyle {
    private final static String[] vecStrWeekDayNames = getWeekDayNames();
```

```
    private static String[] getWeekDayNames() {
        String[] vec = new String[10];
        vec[Calendar.SUNDAY] = "周日";
        vec[Calendar.MONDAY] = "周一";
        vec[Calendar.TUESDAY] = "周二";
        vec[Calendar.WEDNESDAY] = "周三";
        vec[Calendar.THURSDAY] = "周四";
        vec[Calendar.FRIDAY] = "周五";
        vec[Calendar.SATURDAY] = "周六";
        return vec;
    }
    public static String getWeekDayName(int iDay) {
        return vecStrWeekDayNames[iDay];
    }
    public static int getWeekDay(int index, int iFirstDayOfWeek) {
        int iWeekDay = -1;
        if (iFirstDayOfWeek == Calendar.MONDAY) {
            iWeekDay = index + Calendar.MONDAY;
            if (iWeekDay > Calendar.SATURDAY)
                iWeekDay = Calendar.SUNDAY;
        }
        if (iFirstDayOfWeek == Calendar.SUNDAY) {
            iWeekDay = index + Calendar.SUNDAY;
        }
        return iWeekDay;
    }
}
```

运行程序,效果如图 10-5 所示。单击界面中的数字可改变日期,单击界面中的"上一页"按钮可翻转到上一月;单击界面中的"下一页"按钮可翻转到下一月。



图 10-5 万年日历界面

10.1.7 手电筒设计

安卓节能手电筒是完全免费的安卓手机软件,软件是利用手机 Led 闪光灯发光做手电筒的。安卓节能手电筒之所以叫"节能"是因为软件支持"定时关闭"Led 闪光灯(其他手电筒软件不具备该功能)和 Led 闪光灯发出的亮度适中,亮度不会太暗,也不会太亮,在保证照明的情况下保护手机 Led 闪光灯和节省电池电量。

下面通过一个案例来演示 Android 手机的手电筒实现。其具体操作步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 Flash_light。

(2) 打开 res\layout 目录下的 main.xml 布局文件,其代码为:

```
<Relativelayout xmlns:tools = "http://schemas.android.com/tools"
    xmlns:android = "http://schemas.android.com/apk/res/android"
    tools:context = ".MainActivity"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
```

```
    android:gravity = "center"
    android:background = " # aabbcc">
<fs.flash_light.LightBkView
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:id = "@ + id/light1">
  </fs.flash_light.LightBkView>
  </Relativelayout>
```

(3) 打开 src\fs.flash_light 包下的 MainActivity.java 文件,定义单击事件,其代码为:

```
package fs.flash_light;
import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.Menu;
public class MainActivity extends Activity {
    private LightBkView light1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        light1 = (LightBkView) findViewById(R.id.light1);
        //定义单击事件
        light1.setOnClickListener(light1);
        }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

(4) 在 src\fs.flash_light 包下创建一个 LightBkView.java 文件,该文件实现手电筒功能,其代码为:

```
package fs.flash_light;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.hardware.Camera;
import android.hardware.Camera.Parameters;
import android.util.AttributeSet;
import android.view.View;
import android.view.View.OnClickListener;
public class LightBkView extends View implements OnClickListener {
    Camera camera = Camera.open();
    //定义画笔
    Paint paint = new Paint();
    Paint paint1 = new Paint();
    int x = 0;
    int y = 0;
    //打开闪光灯
    boolean islight;
    public LightBkView(Context context, AttributeSet set) {
        super(context, set);
```

```
    }
@Override
protected void onDraw(Canvas canvas) {
    //获取控件的宽度和高度
    int width = this.getWidth();
    int heigth = this.getHeight();
    //圆点的坐标
    x = width / 2;
    y = heigth / 2;
    //更换开关背景
    if(!islight){
        paint.setColor(Color.BLUE);
        canvas.drawCircle(x, y, 60, paint);
        paint1.setColor(Color.RED);
        paint1.setTextSize(20);
        canvas.drawText("打开闪光灯", x - 50, y, paint1);
        invalidate();
        }else{
            paint.setColor(Color.WHITE);
            canvas.drawCircle(x, y, 60, paint);
            paint1.setColor(Color.RED);
            paint1.setTextSize(20);
            canvas.drawText("关闭闪光灯", x - 50, y, paint1);
            invalidate();
            }
    }
//定义 View 的大小
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    setMeasuredDimension(getWidth(widthMeasureSpec), getHeight(heightMeasureSpec));
    }
//定义 view 的宽度
public int getWidth(int widthMeasureSpec) {
    int reslut = 0;
    int widthMode = MeasureSpec.getMode(widthMeasureSpec);
    if (widthMode == MeasureSpec.AT_MOST) {
        reslut = 120;
        }
    if (widthMode == MeasureSpec.EXACTLY) {
        reslut = MeasureSpec.getSize(widthMeasureSpec);
        }
    return reslut;
    }
//定义 view 的高度
public int getHeight(int heightMeasureSpec) {
    int reslut = 0;
    int heightMode = MeasureSpec.getMode(heightMeasureSpec);
    if (heightMode == MeasureSpec.AT_MOST) {
        reslut = 120;
        }
    if (heightMode == MeasureSpec.EXACTLY) {
        reslut = MeasureSpec.getSize(heightMeasureSpec);
        }          return reslut;
        }
//实现闪光灯的开关
```

```
        @Override
        public void onClick(View v) {
            if (!islight) {
                Parameters mParameters = camera.getParameters();
                mParameters.setFlashMode(Camera.Parameters.FLASH_MODE_TORCH);
                camera.setParameters(mParameters);
                islight = true;
                } else {
                    Parameters mParameters = camera.getParameters();

                    mParameters.setFlashMode(Camera.Parameters.FLASH_MODE_OFF);
                    camera.setParameters(mParameters);
                    islight = false;
                    }
            }
    }
```

(5) 打开 AndroidManifest.xml 项目文件,设置手电筒权限,其代码为:

```
⋮
    </application>
    <!-- 摄像头、手电筒 -->
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.FLASHLIGHT" />
    <uses-feature android:name="android.hardware.camera" />
    <uses-feature android:name="android.hardware.camera.autofocus" />
     <uses-feature android:name="android.hardware.camera.flash" />
</manifest>
```

10.1.8 计算器实现

计算器是当今所有手机上都集成的功能,Android 手机也不例外。Android 手机的计算器功能越来越多,使用也越来越方便。下面通过一个案例来演示计算器界面的生成。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 Calculators_Manager。

(2) 打开 res\layout 目录下的 main.xml 文件,在文件中实现相应的控件声明和计算器界面,其代码为:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="#ffcc66">
    <LinearLayout android:layout_width="fill_parent"
    android:layout_height="wrap_content">
  <TextView
      android:id="@+id/tvResult"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:height="50dp"
      android:text="" />
</LinearLayout>
<LinearLayout android:layout_width="fill_parent"
    android:layout_height="wrap_content">
```

```
    <Button
        android:id="@+id/btnBackspace"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:width="150dp"
        android:layout_marginLeft="10dp"
        android:text="backspace"/>
      <Button
        android:id="@+id/btnCE"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:width="150dp"
        android:text="CE"/>
</LinearLayout>
<LinearLayout android:layout_width="fill_parent"
    android:layout_height="wrap_content">
      <Button
        android:id="@+id/btn7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:width="75dp"
        android:text="7"/>
       <Button
        android:id="@+id/btn8"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:width="75dp"
        android:text="8"/>
      <Button
        android:id="@+id/btn9"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:width="75dp"
        android:text="9"/>
      <Button
        android:id="@+id/btnDiv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:width="75dp"
        android:text="/"/>
</LinearLayout>
<LinearLayout android:layout_width="fill_parent"
    android:layout_height="wrap_content">
      <Button
        android:id="@+id/btn4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:width="75dp"
        android:text="4"/>
      <Button
        android:id="@+id/btn5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
            android:width = "75dp"
            android:text = "5"/>
          < Button
            android:id = "@ + id/btn6"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:width = "75dp"
            android:text = "6"/>
          < Button
            android:id = "@ + id/btnMul"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:width = "75dp"
            android:text = " * "/>
    </LinearLayout >
    < LinearLayout android:layout_width = "fill_parent"
        android:layout_height = "wrap_content">
          < Button
            android:id = "@ + id/btn1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_marginLeft = "10dp"
            android:width = "75dp"
            android:text = "1"/>
         < Button
            android:id = "@ + id/btn2"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:width = "75dp"
            android:text = "2"/>
         < Button
            android:id = "@ + id/btn3"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:width = "75dp"
            android:text = "3"/>
          < Button
            android:id = "@ + id/btnAdd"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:width = "75dp"
            android:text = " + "/>
    </LinearLayout >
    < LinearLayout android:layout_width = "fill_parent"
        android:layout_height = "wrap_content">
          < Button
            android:id = "@ + id/btn0"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_marginLeft = "10dp"
            android:width = "75dp"
            android:text = "0"/>
         < Button
            android:id = "@ + id/btnC"
            android:layout_width = "wrap_content"
```

```
        android:layout_height = "wrap_content"
        android:width = "75dp"
        android:text = "C"/>
    < Button
        android:id = "@ + id/btnEqu"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:width = "75dp"
        android:text = " = "/>
    < Button
        android:id = "@ + id/btnSub"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:width = "75dp"
        android:text = " - "/>
</LinearLayout >
</LinearLayout >
```

（3）打开 src\fs. calculators_manager 包下的 MainActivity. java 文件，在文件中实现计算器功能，其代码为：

```
package fs.calculators_manager;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.app.Activity;
public class MainActivity extends Activity  implements OnClickListener{
    //声明一些控件
    Button btn0 = null;
    Button btn1 = null;
    Button btn2 = null;
    Button btn3 = null;
    Button btn4 = null;
    Button btn5 = null;
    Button btn6 = null;
    Button btn7 = null;
    Button btn8 = null;
    Button btn9 = null;
    Button btnBackspace = null;
    Button btnCE = null;
    Button btnC = null;
    Button btnAdd = null;
    Button btnSub = null;
    Button btnMul = null;
    Button btnDiv = null;
    Button btnEqu = null;
    TextView tvResult = null;
    //声明两个参数,接收 tvResult 前后的值
    double num1 = 0,num2 = 0;
    double Result = 0;                                  //计算结果
    int op = 0;                                         //判断操作数,
    boolean isClickEqu = false;                         //判断是否按" = "键
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //从布局文件中获取控件
        btn0 = (Button)findViewById(R.id.btn0);
        btn1 = (Button)findViewById(R.id.btn1);
        btn2 = (Button)findViewById(R.id.btn2);
        btn3 = (Button)findViewById(R.id.btn3);
        btn4 = (Button)findViewById(R.id.btn4);
        btn5 = (Button)findViewById(R.id.btn5);
        btn6 = (Button)findViewById(R.id.btn6);
        btn7 = (Button)findViewById(R.id.btn7);
        btn8 = (Button)findViewById(R.id.btn8);
        btn9 = (Button)findViewById(R.id.btn9);
        btnBackspace = (Button)findViewById(R.id.btnBackspace);
        btnCE = (Button)findViewById(R.id.btnCE);
        btnC = (Button)findViewById(R.id.btnC);
        btnEqu = (Button)findViewById(R.id.btnEqu);
        btnAdd = (Button)findViewById(R.id.btnAdd);
        btnSub = (Button)findViewById(R.id.btnSub);
        btnMul = (Button)findViewById(R.id.btnMul);
        btnDiv = (Button)findViewById(R.id.btnDiv);
        tvResult = (TextView)findViewById(R.id.tvResult);
        //添加监听
        btnBackspace.setOnClickListener(this);
        btnCE.setOnClickListener(this);
        btn0.setOnClickListener(this);
        btn1.setOnClickListener(this);
        btn2.setOnClickListener(this);
        btn3.setOnClickListener(this);
        btn4.setOnClickListener(this);
        btn5.setOnClickListener(this);
        btn6.setOnClickListener(this);
        btn7.setOnClickListener(this);
        btn8.setOnClickListener(this);
        btn9.setOnClickListener(this);
        btnAdd.setOnClickListener(this);
        btnSub.setOnClickListener(this);
        btnMul.setOnClickListener(this);
        btnDiv.setOnClickListener(this);
        btnEqu.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
        //btnBackspace 和 CE --------------------
        case R.id.btnBackspace:
            String myStr = tvResult.getText().toString();
            try {
                tvResult.setText(myStr.substring(0, myStr.length() - 1));
            } catch (Exception e) {
                tvResult.setText("");
            }
            break;
        case R.id.btnCE:
            tvResult.setText(null);
```

```
        break;
        //btn0～9 ---------------------------
    case R.id.btn0:
        if(isClickEqu)
        {
            tvResult.setText(null);
            isClickEqu = false;
        }
        String myString = tvResult.getText().toString();
        myString += "0";
        tvResult.setText(myString);
        break;
    case R.id.btn1:
        if(isClickEqu)
        {
            tvResult.setText(null);
            isClickEqu = false;
        }
        String myString1 = tvResult.getText().toString();
        myString1 += "1";
        tvResult.setText(myString1);
        break;
    case R.id.btn2:
        if(isClickEqu)
        {
            tvResult.setText(null);
            isClickEqu = false;
        }
        String myString2 = tvResult.getText().toString();
        myString2 += "2";
        tvResult.setText(myString2);
        break;
    case R.id.btn3:
        if(isClickEqu)
        {
            tvResult.setText(null);
            isClickEqu = false;
        }
        String myString3 = tvResult.getText().toString();
        myString3 += "3";
        tvResult.setText(myString3);
        break;
    case R.id.btn4:
        if(isClickEqu)
        {
            tvResult.setText(null);
            isClickEqu = false;
        }
        String myString4 = tvResult.getText().toString();
        myString4 += "4";
        tvResult.setText(myString4);
        break;
    case R.id.btn5:
        if(isClickEqu)
        {
```

```
            tvResult.setText(null);
            isClickEqu = false;
        }
        String myString5 = tvResult.getText().toString();
        myString5 += "5";
        tvResult.setText(myString5);
        break;
    case R.id.btn6:
        if(isClickEqu)
        {
            tvResult.setText(null);
            isClickEqu = false;
        }
        String myString6 = tvResult.getText().toString();
        myString6 += "6";
        tvResult.setText(myString6);
        break;
    case R.id.btn7:
        if(isClickEqu)
        {
            tvResult.setText(null);
            isClickEqu = false;
        }
        String myString7 = tvResult.getText().toString();
        myString7 += "7";
        tvResult.setText(myString7);
        break;
    case R.id.btn8:
        if(isClickEqu)
        {
            tvResult.setText(null);
            isClickEqu = false;
        }
        String myString8 = tvResult.getText().toString();
        myString8 += "8";
        tvResult.setText(myString8);
        break;
    case R.id.btn9:
        if(isClickEqu)
        {
            tvResult.setText(null);
            isClickEqu = false;
        }
        String myString9 = tvResult.getText().toString();
        myString9 += "9";
        tvResult.setText(myString9);
        break;
        //btn+- * / = --------------------------------
    case R.id.btnAdd:
        String myStringAdd = tvResult.getText().toString();
        if(myStringAdd.equals(null))
        {
            return;
        }
        num1 = Double.valueOf(myStringAdd);
```

```
        tvResult.setText(null);
        op = 1;
        isClickEqu = false;
        break;
    case R.id.btnSub:
        String myStringSub = tvResult.getText().toString();
        if(myStringSub.equals(null))
        {
            return;
        }
        num1 = Double.valueOf(myStringSub);
        tvResult.setText(null);
        op = 2;
        isClickEqu = false;
        break;
    case R.id.btnMul:
        String myStringMul = tvResult.getText().toString();
        if(myStringMul.equals(null))
        {
            return;
        }
        num1 = Double.valueOf(myStringMul);
        tvResult.setText(null);
        op = 3;
        isClickEqu = false;
        break;
    case R.id.btnDiv:
        String myStringDiv = tvResult.getText().toString();
        if(myStringDiv.equals(null))
        {
            return;
        }
        num1 = Double.valueOf(myStringDiv);
        tvResult.setText(null);
        op = 4;
        isClickEqu = false;
        break;
    case R.id.btnEqu:
        String myStringEqu = tvResult.getText().toString();
        if(myStringEqu.equals(null))
        {
            return;
        }
        num2 = Double.valueOf(myStringEqu);
        tvResult.setText(null);
        switch (op) {
        case 0:
            Result = num2;
            break;
        case 1:
            Result = num1 + num2;
            break;
        case 2:
            Result = num1 - num2;
            break;
```

```
                case 3:
                    Result = num1 * num2;
                    break;
                case 4:
                    Result = num1/num2;
                    break;
                default:
                    Result = 0;
                    break;
                }
                tvResult.setText(String.valueOf(Result));
                isClickEqu = true;
                break;
            default:
                break;
            }
        }
    }
```

运行程序，效果如图 10-6 所示。



图 10-6　计算器

10.1.9　WiFi 开发

WiFi 就是一种无线联网技术，常见的是使用无线路由器。那么在这个无线路由器信号覆盖的范围内都可以采用 WiFi 连接的方式进行联网。如果无线路由器连接了一个 ADSL 线路或其他的联网线路，则又被称为"热点"。

Android 本身提供了一些有用的包，在 android.net.wifi 包下可以对 WiFi 操作。主要包括以下几个类和接口：

- ScanResult：用来描述已经检测出的接入点，包括接入点的地址、接入点的名称、身份认证、频率、信号强度等信息。
- WifiConfiguration：WiFi 网络的配置，包括安全设置等。
- WifiInfo：WiFi 无线连接的描述，包括接入点、网络连接状态、隐藏的接入点、IP 地址、连接速度、MAC 地址、网络 ID、信号强度等信息。这里简单介绍一下这些方法：

getBSSID()：获取 BSSID。

getDetailedStateOf()：获取客户端的连通性。

getHiddenSSID()：获得 SSID 是否被隐藏。

getIpAddress()：获取 IP 地址。

getLinkSpeed()：获得连接的速度。

getMacAddress()：获得 MAC 地址。

getRssi()：获得 802.11n 网络的信号。

getSSID()：获得 SSID。

getSupplicanState()：返回具体客户端状态的信息。

• WifiManager：用来管理 WiFi 连接，这里已经定义好了一些类，可以供使用。

下面通过一个案例来实现手机上的 WiFi。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Wi_Fi_Manager。

(2) 打开 res\layout 目录下的 main.xml，声明一个 ScrollView 控件，在控件中定义一个线性布局，在线性布局中声明 4 个 Button 控件及一个 TextView 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< ScrollView xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = " # ffcc66">
 < LinearLayout
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = " # ffcc66">
< Button
    android:id = "@ + id/scan"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "扫描网络"/>
< Button
    android:id = "@ + id/start"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "打开 WiFi"/>
< Button
    android:id = "@ + id/stop"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "关闭 WiFi"/>
< Button
    android:id = "@ + id/check"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "WiFi 状态"/>
  < TextView
    android:id = "@ + id/allNetWork"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "当前没有扫描到 WiFi 网络"/>
    </LinearLayout >
</ScrollView >
```

(3) 打开 src\wi_fi_manager 包下的 MainActivity.java 文件，该文件用于实现 WiFi 扫描、打开、关闭等监听功能，其代码为：

```
package fs.wi_fi_manager;
import java.util.List;
import android.app.Activity;
import android.net.wifi.ScanResult;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
```

```
import android.widget.TextView;
import android.widget.Toast;
public class MainActivity extends Activity {
    /** 第一次调用活动 */
      private TextView allNetWork;
        private Button scan;
        private Button start;
        private Button stop;
        private Button check;
        private WifiAdmin mWifiAdmin;
        //扫描结果列表
        private List<ScanResult> list;
        private ScanResult mScanResult;
        private StringBuffer sb = new StringBuffer();
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mWifiAdmin = new WifiAdmin(MainActivity.this);
        init();
    }
    public void init(){
        allNetWork = (TextView) findViewById(R.id.allNetWork);
        scan = (Button) findViewById(R.id.scan);
        start = (Button) findViewById(R.id.start);
        stop = (Button) findViewById(R.id.stop);
        check = (Button) findViewById(R.id.check);
        scan.setOnClickListener(new MyListener());
        start.setOnClickListener(new MyListener());
        stop.setOnClickListener(new MyListener());
        check.setOnClickListener(new MyListener());
    }
    private class MyListener implements OnClickListener{
        @Override
        public void onClick(View v) {
            //TODO Auto-generated method stub
            switch (v.getId()) {
            case R.id.scan:                                     //扫描网络
                  getAllNetWorkList();
                break;
           case R.id.start:                                     //打开 WiFi
                mWifiAdmin.openWiFi();
                Toast.makeText(MainActivity.this, "当前 WiFi 状态为: " + mWifiAdmin.checkState(),
1).show();
                break;
           case R.id.stop:                                      //关闭 WiFi
                mWifiAdmin.closeWiFi();
                Toast.makeText(MainActivity.this, "当前 WiFi 状态为: " + mWifiAdmin.checkState(),
1).show();
                break;
           case R.id.check:                                     //WiFi 状态
               Toast.makeText(MainActivity.this, "当前 WiFi 状态为: " + mWifiAdmin.checkState(),
1).show();
                break;
            default:
```

```
                break;
            }
        }
    }
    public void getAllNetWorkList(){
          //每次单击扫描之前清空上一次的扫描结果
        if(sb!= null){
            sb = new StringBuffer();
        }
        //开始扫描网络
        mWifiAdmin.startScan();
        list = mWifiAdmin.getWifiList();
        if(list!= null){
            for(int i = 0;i < list.size();i++){
                //得到扫描结果
                mScanResult = list.get(i);
           sb = sb.append(mScanResult.BSSID + "   ").append(mScanResult.SSID + "   ")
        .append(mScanResult.capabilities + "   ").append(mScanResult.frequency + "   ")
                .append(mScanResult.level + "\n\n");
            }
            allNetWork.setText("扫描到的 WiFi 网络: \n" + sb.toString());
        }
    }
}
```

(4) 接下来把 WiFi 相关操作都封装在一个 WifiAdmin 类中,以后开启或关闭等相关操作可以直接调用这个类的相关方法。在 src\wi_fi_manager 包下创建一个 WifiAdmin.java 文件,其代码为:

```
package fs.wi_fi_manager;
import java.util.List;
import android.content.Context;
import android.net.wifi.ScanResult;
import android.net.wifi.WifiConfiguration;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.net.wifi.WifiManager.WifiLock;
public class WifiAdmin {
    //定义一个 WifiManager 对象
    private WifiManager mWifiManager;
    //定义一个 WifiInfo 对象
    private WifiInfo mWifiInfo;
    //扫描出的网络连接列表
    private List < ScanResult > mWifiList;
    //网络连接列表
    private List < WifiConfiguration > mWifiConfigurations;
    WifiLock mWifiLock;
    public WifiAdmin(Context context){
        //取得 WifiManager 对象
        mWifiManager = (WifiManager) context.getSystemService(Context.WIFI_SERVICE);
        //取得 WifiInfo 对象
        mWifiInfo = mWifiManager.getConnectionInfo();
    }
    //打开 WiFi
    public void openWifi(){
        if(!mWifiManager.isWifiEnabled()){
```

```
            mWifiManager.setWifiEnabled(true);
        }
    }
    //关闭 WiFi
    public void closeWifi(){
        if(!mWifiManager.isWifiEnabled()){
            mWifiManager.setWifiEnabled(false);
        }
    }
     //检查当前 WiFi 状态
    public int checkState() {
        return mWifiManager.getWifiState();
    }
    //锁定 wifiLock
    public void acquireWifiLock(){
        mWifiLock.acquire();
    }
    //解锁 wifiLock
    public void releaseWifiLock(){
        //判断是否锁定
        if(mWifiLock.isHeld()){
            mWifiLock.acquire();
        }
    }
    //创建一个 wifiLock
    public void createWifiLock(){
        mWifiLock = mWifiManager.createWifiLock("test");
    }
    //得到配置好的网络
    public List<WifiConfiguration> getConfiguration(){
        return mWifiConfigurations;
    }
    //指定配置好的网络进行连接
    public void connetionConfiguration(int index){
        if(index > mWifiConfigurations.size()){
            return;
        }
        //连接配置好指定 ID 的网络
        mWifiManager.enableNetwork(mWifiConfigurations.get(index).networkId, true);
    }
    public void startScan(){
        mWifiManager.startScan();
        //得到扫描结果
        mWifiList = mWifiManager.getScanResults();
        //得到配置好的网络连接
        mWifiConfigurations = mWifiManager.getConfiguredNetworks();
    }
    //得到网络列表
    public List<ScanResult> getWifiList(){
        return mWifiList;
    }
    //查看扫描结果
    public StringBuffer lookUpScan(){
        StringBuffer sb = new StringBuffer();
        for(int i = 0;i < mWifiList.size();i++){
```

```
            sb.append("Index_" + new Integer(i + 1).toString() + ":");
            //将 ScanResult 信息转换成一个字符串包
            //其中包括 BSSID、SSID、capabilities、frequency、level
            sb.append((mWifiList.get(i)).toString()).append("\n");
        }
        return sb;
    }
    public String getMacAddress(){
        return (mWifiInfo == null)?"NULL":mWifiInfo.getMacAddress();
    }
    public String getBSSID(){
        return (mWifiInfo == null)?"NULL":mWifiInfo.getBSSID();
    }
    public int getIpAddress(){
        return (mWifiInfo == null)?0:mWifiInfo.getIpAddress();
    }
    //得到连接的 ID
    public int getNetWordId(){
        return (mWifiInfo == null)?0:mWifiInfo.getNetworkId();
    }
    //得到 wifiInfo 的所有信息
    public String getWifiInfo(){
        return (mWifiInfo == null)?"NULL":mWifiInfo.toString();
    }
    //添加一个网络并连接
    public void addNetWork(WifiConfiguration configuration){
        int wcgId = mWifiManager.addNetwork(configuration);
        mWifiManager.enableNetwork(wcgId, true);
    }
    //断开指定 ID 的网络
    public void disConnectionWifi(int netId){
        mWifiManager.disableNetwork(netId);
        mWifiManager.disconnect();
    }
}
```

(5) 打开 AndroidManifest 项目配置文件,在文件中添加权限,添加代码为:

```
⋮
    </application>
<!-- 以下是使用 WiFi 访问网络所需的权限 -->
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
 <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
 <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
</manifest>
```

运行程序,默认界面如图 10-7(a)所示。当单击相关按钮,即弹出对应的 Toast 消息,效果如图 10-7(b)所示。

10.1.10 通讯录搜索

本节将要介绍怎样对通讯录中联系人进行搜索,主要是对 ContentResolver 的应用。

手机中的通讯录是一个不可缺少的部分,用户的所有联系人均在里面,本软件实现通过自制的手机通讯录软件对联系人进行简单搜索。



(a) 默认界面　　(b) Toast消息提示

图 10-7　WiFi 实现

在 EditText 中输入所要搜索联系人名字的首字母，在自定义的 ContentResolver 中会显示首字母相同的联系人，当输入的是“*”时，则会显示所有的联系人。当单击其中一个联系人时，会在主界面中显示该联系人的姓名和电话。如果该用户没有电话，则会显示该联系人无电话号码。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Addressbook_Search。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明一个 TextView 控件及一个 AutoCompleteTextViev 控件。

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aaaccc">
    <AutoCompleteTextView
     android:id = "@ + id/AutoCompleteTextView1"
     android:layout_width = "fill_parent"
     android:layout_height = "wrap_content"/>
    <TextView
     android:id = "@ + id/TextView1"
     android:layout_width = "fill_parent"
     android:layout_height = "wrap_content"
     android:textColor = "#222333"/>
</RelativeLayout>
```

(3) 打开 src\fs.addressbook_search 包下的 MainActivity.java 文件，在文件中声明查询的通讯录字段名，并显示搜索结果，其代码为：

```java
import android.app.Activity;
import android.content.ContentResolver;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.Contacts;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AutoCompleteTextView;
import android.widget.TextView;
public class MainActivity extends Activity {
    private AutoCompleteTextView Aclt;                              //声明引用
    private TextView tv;
    private Cursor cursor;
    private ContactsAdapter Ca;
    static final String[] PEOPLE_PROJECTION =
    {
        Contacts.People._ID,
        Contacts.People.PRIMARY_PHONE_ID,
        Contacts.People.TYPE,
        Contacts.People.NUMBER,
        Contacts.People.LABEL,
        Contacts.People.NAME
    };
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
Aclt = (AutoCompleteTextView)this.findViewById(R.id.AutoCompleteTextView1);  //获取对象
        tv = (TextView)this.findViewById(R.id.TextView1);
        ContentResolver content = getContentResolver();             //获取 ContentResolver 对象
        //取得通讯录的 Cursor
        cursor = content.query
        (
                Contacts.People.CONTENT_URI,
                PEOPLE_PROJECTION,
                null,
                null,
                Contacts.People.DEFAULT_SORT_ORDER
        );
        Ca = new ContactsAdapter(this,cursor);                       //创建对象
        Aclt.setAdapter(Ca);
        Aclt.setOnItemClickListener
        (
            new AdapterView.OnItemClickListener()
            {
                public void onItemClick(AdapterView<?> arg0, View arg1,
                        int arg2, long arg3) {
                    Cursor c = Ca.getCursor();                       //取得 Cursor 对象
                    c.moveToPosition(arg2);                          //移动到点击位置
                    String number = c.getString(c.getColumnIndexOrThrow(Contacts.People.NUMBER));
                                                                     //获取电话号码
                    if(number == null)
                    {
                        number = "该联系人无电话号码";
                    }
```

```
                String name = c.getString(c.getColumnIndexOrThrow(Contacts.People.NAME));
            tv.setText("联系人姓名: " + name + ",联系人电话: " + number + ".");
                }
            }
        );
    }
}
```

（4）在 src\fs.addressbook_search 包下创建一个 ContactsAdapter 类，在该类中获取通讯录中联系人的姓名，并获取搜索字符为"*"时的结果集。其代码为：

```
import android.content.ContentResolver;
import android.content.Context;
import android.database.Cursor;
import android.provider.Contacts;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CursorAdapter;
import android.widget.TextView;
public class ContactsAdapter extends CursorAdapter{
    ContentResolver Cr;
    public ContactsAdapter(Context context, Cursor c) {
        super(context, c);
        Cr = context.getContentResolver();
    }
    @Override
    public void bindView(View arg0, Context arg1, Cursor arg2) {
        //获取通讯录中的姓名
        ((TextView) arg0).setText(arg2.getString
(arg2.getColumnIndexOrThrow(Contacts.People.NAME)));
    }
    @Override
    public View newView(Context arg0, Cursor arg1, ViewGroup arg2)
    {
        final LayoutInflater myLi = LayoutInflater.from(arg0);
        final TextView tv = (TextView)myLi.inflate
        (
android.R.layout.simple_dropdown_item_1line,arg2,false);
        tv.setText                                          //设置显示文字
        (
            arg1.getString(arg1.getColumnIndexOrThrow
(Contacts.People.NAME))
        );
        return tv;
    }
    @Override
    public String convertToString(Cursor cursor)
    {
        String str = cursor.getString
(cursor.getColumnIndexOrThrow(Contacts.People.NAME));
        return str;
    }
    @Override
    public Cursor runQueryOnBackgroundThread(CharSequence cs)
    {
```

```
        if(getFilterQueryProvider()!= null)
        {
            return getFilterQueryProvider().runQuery(cs);
        }
        StringBuilder sb = new StringBuilder();
        String[] str = null;
        if(cs!= null)
        {
            sb.append("UPPER(");
            sb.append(Contacts.People.NAME);
            sb.append(") GLOB ?");
            str = new String[ ]
               {
                    cs.toString().toUpperCase() + " * "
               };
        }
        //返回搜索结果
        return Cr.query(
                Contacts.People.CONTENT_URI,
                MainActivity.PEOPLE_PROJECTION,
                sb == null?null:sb.toString(),
                str,
                Contacts.People.DEFAULT_SORT_ORDER
                );
    }
}
```

(5) 打开 AndroidManifest.xml 文件,设置通讯录搜索权限,其代码为:

```
⋮
  </application>
<!-- 添加通讯录搜索权限 -->
    <uses-permission android:name = "android.permission.READ_CONTACTS"></uses-permission>
</manifest>
```

运行程序,效果如图 10-8 所示。



图 10-8　通讯录搜索界面

10.2 通话功能

语音通话是手机设备最基本也是最重要的功能，在 Android 系统中不仅可以统计通话号码、通话时间等基本信息，还可以方便地拨出号码、自动挂断、接通电话以及电话录音等。

10.2.1 拨打电话

手机最基本的功能就是拨打电话。在具体拨打电话时，首先要在 AndroidManifest 中添加 uses-permission，这样就实现了对拨打电话的声明；然后通过自定义的 Intent 对象、ACTION_CALL 键和 Uri.parse()方法将用户输入的电话号码写入；最后，通过 startActivity 方法即可完成程序拨打电话的功能。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Phone_Call。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 TextView 控件、一个 Button 控件及一个 EditText 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#ffcc66">
<TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "请输入电话号码"/>
<EditText
    android:id = "@ + id/phonenumber"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:phoneNumber = "true" />
 <Button
    android:id = "@ + id/btn_call"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "拨打电话" />
</LinearLayout>
```

(3) 打开 src\fs.phone_call 包下的 MainActivity.java 文件，在文件中实现电话的拨打，其代码为：

```
package fs.phone_call;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
public class MainActivity extends Activity {
    /** 第一次调用活动. */
```

```
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            Button btn_call = (Button) findViewById(R.id.btn_call);
            btn_call.setOnClickListener(new OnClickListener() {
                public void onClick(View v) {
                    //TODO 自动存根法
                    EditText et_phonenumber = (EditText)findViewById(R.id.phonenumber);
                    String number = et_phonenumber.getText().toString();
                    //用 intent 启动拨打电话
                    Intent intent = new Intent(Intent.ACTION_CALL,Uri.parse("tel:" + number));
                    startActivity(intent);
                }
            });
        }
    }
```

(4) 打开 AndroidManifest.xml 项目配置文件,添加电话权限,其代码为:

```
    ⋮
        <uses-sdk
            android:minSdkVersion = "8"
            android:targetSdkVersion = "18" />
        <!-- 添加向外拨打电话的权限 -->
        <uses-permission android:name = "android.permission.CALL_PHONE"/>
        <application
            android:allowBackup = "true"
            android:icon = "@drawable/ic_launcher"
            android:label = "@string/app_name"
            android:theme = "@style/AppTheme" >
    ⋮
```

运行程序,界面如图 10-9 所示。



图 10-9 拨打电话界面

10.2.2 来电黑名单

在 Android 手机上实现一个类似“来电黑名单”的功能,如果遇到黑名单中的号码来电,系统就会提示用户选择接听或挂断。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 Call_Blacklist。

(2) 在 res\layout 目录下创建一个 choice.xml 文件,该文件用于删除、添加黑名单界面,其代码为:

```
<?xml version = "1.0" encoding = "utf-8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:orientation = "vertical"
    android:background = "#ffcc66">
```

```
    <EditText
        android:id = "@ + id/editText1"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:hint = "请输入添加的黑名单号码" >
    </EditText >
    <LinearLayout
        android:layout_width = "match_parent"
        android:layout_height = "match_parent">
        <Button
            android:id = "@ + id/add"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "添加" />
        <Button
            android:id = "@ + id/del"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "删除" />
    </LinearLayout >
</LinearLayout >
```

(3) 在 res\layout 目录下创建一个 result. xml 文件,用于实现查看结果界面,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:orientation = "vertical"
    android:background = "#ffcc66">
    <ListView
        android:id = "@ + id/listView1"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content">
    </ListView >
</LinearLayout >
```

(4) 打开 res\layout 目录下的 main. xml 文件,用于实现主界面,在文件中声明一个 Button 控件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:gravity = "center_horizontal"
    android:background = "#ffcc66">
<Button
    android:id = "@ + id/managerBlock"
    android:singleLine = "true"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "管理黑名单" />
</LinearLayout >
```

(5) 打开 src\call_blacklist 包下的 MainActivity. java 文件,在文件中实现事件绑定,其代码为:

```
package fs.call_blacklist;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class MainActivity extends Activity {
    /** 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btn1 = (Button)findViewById(R.id.managerBlock);
        //绑定事件
        btn1.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View arg0) {
                //TODO 自动存根法
                Intent intent = new Intent(MainActivity.this, Result.class);
                startActivity(intent);
            }
        });
    }
}
```

(6) 在 src\call_blacklist 包下创建一个 Result.java 文件,文件实现黑名单列表,其代码为:

```
import java.util.ArrayList;
import java.util.List;
import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
//此 Activity 用于显示 LISTVIEW 即黑名单列表
public class Result extends Activity {
    DatabaseHelper dbHelper;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //TODO 自动存根法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.result);
        dbHelper = new DatabaseHelper(this, "mydatabase.db3", null, 1);
        final ListView listview = (ListView)findViewById(R.id.listView1);
        listview.setAdapter ( new ArrayAdapter < String > ( this, android.R.layout.simple_
expandable_list_item_1,getData()));
        if(getData().isEmpty()){
            Toast.makeText(Result.this, "该黑名单中无号码,请添加", 5000).show();
            Intent intent = new Intent(Result.this,Choice.class);
            intent.putExtra("phone", "kong");
```

```
            startActivity(intent);
            finish();
        }
        listview.setOnItemClickListener(new OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                    long arg3) {
                //TODO 自动存根法
                Intent intent = new Intent(Result.this,Choice.class);
                intent.putExtra("phone",arg0.getItemAtPosition(arg2).toString());
                startActivity(intent);
                finish();
            }
        });
    }
    private List<String> getData(){
        List<String> data = new ArrayList<String>();
        Cursor cursor = dbHelper.getReadableDatabase().rawQuery(
                "select * from List",null);
        while(cursor.moveToNext()){
            data.add(cursor.getString(1));
        }
        return data;
    }
}
```

(7) 在 src\call_blacklist 包下创建一个 DatabaseHelper.java 文件，文件用于创建一个列表表格，其代码为：

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
public class DatabaseHelper extends SQLiteOpenHelper{
final String CREATE_TABLE_SQL = "创建一个列表表格(_id integer primary key autoincrement,
phone)";
public DatabaseHelper(Context context, String name,CursorFactory factory, int version) {
        super(context, name, factory, version);
        //TODO 自动存根法
    }@Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_TABLE_SQL);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        }
}
```

(8) 在 src\call_blacklist 包下创建一个 Choice.java 文件，在文件中黑名单的添加、删除操作，其代码为：

```
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
```

```
import android.widget.EditText;
import android.widget.Toast;
public class Choice extends Activity{
    DatabaseHelper dbHelper;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //TODO 自动存根法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.choice);
        dbHelper = new DatabaseHelper(this, "mydatabase.db3", null, 1);
        Intent intent = getIntent();
        Bundle data = intent.getExtras();
        final String phone = data.getString("phone");
        //将输入电话存入数据库中
        findViewById(R.id.add).setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
    String addPhone = ((EditText)findViewById(R.id.editText1)).getText().toString();
            dbHelper.getReadableDatabase().execSQL("insert into list values(null, ?)",
                        new String[]{addPhone});
                Toast.makeText(Choice.this, "添加成功",5000).show();
                Intent intent = new Intent(Choice.this,Result.class);
                startActivity(intent);
                finish();
            }
        });
        //将指定电话从黑名单中删除
        findViewById(R.id.del).setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                if(phone == "kong"){
                    Toast.makeText(Choice.this, "请添加号码",5000).show();
                }
                else{
                    dbHelper.getReadableDatabase().execSQL("delete from list where phone = ?",
new String[]{phone});
                    Toast.makeText(Choice.this, "删除成功", 5000).show();
                }
                Intent intent = new Intent(Choice.this,Result.class);
                startActivity(intent);
                finish();
            }
        });
    }
}
```

(9) 在 src\call_blacklist 包下创建一个 BroadcastJie.java 文件，在文件中实现电话监听，其代码为：

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
public class BroadcastJie extends BroadcastReceiver{
    @Override
```

```
    public void onReceive(Context context, Intent intent) {
//判断电话状态,如果是拨打电话,什么都不做,否则执行 else,启动其监听的 Service
        if (intent.getAction().equals(
                Intent.ACTION_NEW_OUTGOING_CALL)){
        }
        else{
            Intent Sbintent = new Intent(context,Blacklist.class);
            context.startService(Sbintent);
        }
    }
}
```

(10) 在 src\call_blacklist 包下创建一个 Blacklist.java 文件,在文件中实现判断输入的号码是否为黑名单,从而进行相应的操作,其代码为:

```
import java.lang.reflect.Method;
import android.app.Service;
import android.content.Intent;
import android.database.Cursor;
import android.os.IBinder;
import android.telephony.PhoneStateListener;
import android.telephony.TelephonyManager;
public class Blacklist extends Service {
    DatabaseHelper dbHelper = new DatabaseHelper(this, "mydatabase.db3", null, 1);
    TelephonyManager tManager;
    CustomPhoneCallListener cpListener;
    public class CustomPhoneCallListener extends PhoneStateListener
    {
        @Override
        public void onCallStateChanged(int state, String incomingNumber)
        {
            switch (state)
            {
                case TelephonyManager.CALL_STATE_IDLE:
                    break;
                case TelephonyManager.CALL_STATE_OFFHOOK:
                    break;
                //当电话呼入时
                case TelephonyManager.CALL_STATE_RINGING:
                    //如果该号码属于黑名单
                    System.out.println("this is 1") ;
                    if (isBlock(incomingNumber))
                    {
                        System.out.println("this is 2") ;
                        try
                        {
                            Method method = Class.forName(
                                "android.os.ServiceManager").getMethod(
                                "getService", String.class);
                //获取远程 TELEPHONY_SERVICE 的 IBinder 对象的代理
                            IBinder binder = (IBinder) method.invoke(null,
                                new Object[] { TELEPHONY_SERVICE });
                        }
                        catch (Exception e)
                        {
                            e.printStackTrace();
```

```
                    }
                }
                break;
            }
            super.onCallStateChanged(state, incomingNumber);
        }
        public boolean isBlock(String phone)
        {
            Cursor cursor = dbHelper.getReadableDatabase().rawQuery(
                    "select * from List",null);
            while(cursor.moveToNext()){
                if (cursor.getString(1).equals(phone)){
                    return true;
                }
            }
            return false;
        }
    }
    @Override
    public IBinder onBind(Intent arg0) {
        return null;
    }
    @Override
    public void onStart(Intent intent, int startId) {
        tManager = (TelephonyManager) getSystemService(TELEPHONY_SERVICE);
        cpListener = new CustomPhoneCallListener();
        //通过 TelephonyManager 监听通话状态的改变
        tManager.listen(cpListener, PhoneStateListener.LISTEN_CALL_STATE);
        super.onStart(intent, startId);
    }
}
```

(11) 打开 AndroidManifest.xml 项目配置文件，在文件中添加权限，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.call_blacklist"
    android:versionCode = "1"
    android:versionName = "1.0" >
    <uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
     <uses - permission android:name = "android.permission.CALL_PHONE" />
    <uses - permission android:name = "android.permission.READ_CONTACTS"/>
    <uses - permission android:name = "android.permission.WRITE_CONTACTS"/>
    <uses - permission android:name = "android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses - permission android:name = "android.permission.READ_PHONE_STATE"/>
    <uses - permission android:name = "android.permission.PROCESS_OUTGOING_CALLS"/>
    <application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        <activity
            android:name = "fs.call_blacklist.MainActivity"
            android:label = "@string/app_name" >
            <intent - filter>
```

```
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
                </intent - filter>
        </activity>
        <activity android:name = ".Result"
                android:theme = "@android:style/Theme.Dialog"
                android:label = "黑名单列表">
        </activity>
        <activity android:name = ".Choice"
                android:theme = "@android:style/Theme.Dialog"
                android:label = "请操作">
        </activity>
        <receiver android:name = ".BroadcastJie">
            <intent - filter android:priority = "999">
                <action android:name = "android.intent.action.PHONE_STATE"/>
                    <action android:name = "android.intent.action.NEW_OUTGOING_CALL" />
                <action android:name = "android.intent.action.BOOT_COMPLETED" />
            </intent - filter>
        </receiver>
        <service android:name = ".Blacklist">
        </service>
    </application>
</manifest>
```

运行程序，效果如图 10-10 所示。



图 10-10　黑名单管理

10.3　短信功能

短信作为当今人和人交流中非常重要的方式，珍藏着大家不同时期的心情和成长。

10.3.1　发送短信

Android 中短信主要采用 SmsManager 的 sendTextMessage( ) 方法来发送文字短信，

sendTextMessage()方法有 5 个参数，第 1 个参数为对方的手机号码(不能为空)；第 2 个参数为发送方的手机号号码(可以为空)；第 3 个参数为发送的短信内容(不能为空)；第 4 个参数为 PendingIntent 对象，用于判断发送短信是否成功(可以为空)；第 5 个参数也为 PendingIntent 对象，当用户收到短信时会返回该对象(可以为空)。

下面通过一个案例来演示手机的发送短信功能。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Send_Message。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明两个 TextView 控件、两个 EditText 控件及一个 Button 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#ffcc66">
<TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "号码"/>
<!-- 电话号码输入 -->
<EditText
    android:id = "@ + id/et_phone"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"/>
<TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "短信"/>
<!-- 短信内容编辑 -->
<EditText
    android:id = "@ + id/et_content"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:minLines = "3"
    android:gravity = "left"/>
    <!-- 可 3 行显示 -->
    <!-- 设置左边输入 -->
<Button
    android:id = "@ + id/bt_send"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "发送"/>
</LinearLayout>
```

(3) 打开 src\fs.send_message 包下的 MainActivity.java 文件，在文件中编写发送短信，其代码为：

```
package fs.send_message;
import java.util.List;
import android.app.Activity;
import android.os.Bundle;
```

```
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends Activity {
    private EditText mobileText;
    private EditText contentText;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获取电话号文本框
        mobileText = (EditText)this.findViewById(R.id.et_phone);
        //获取短信内容文本框
        contentText = (EditText)this.findViewById(R.id.et_content);
        //获取按钮
        Button button = (Button)this.findViewById(R.id.bt_send);
        button.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v) {
                //获取电话号码
                String moblie = mobileText.getText().toString();
                //短信内容
                String content = contentText.getText().toString();
                //获取短信管理器
                SmsManager smsManager = SmsManager.getDefault();
                //如果汉字大于 70 个
                if(content.length() > 70){
                    //返回多条短信
                    List < String > contents = smsManager.divideMessage(content);
                    for(String sms:contents){
                          smsManager.sendTextMessage(moblie, null, sms, null, null);
                      }
                    }else{
                            smsManager.sendTextMessage(moblie, null, content, null, null);
                      }
       Toast.makeText(MainActivity.this, "发送成功!", Toast.LENGTH_LONG).show();
                }
            });
    }
}
```

(4) 打开 AndroidManifest.xml 文件配置项目,添加发送短信权限,其代码为:

```
⋮
< uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
   <!-- 添加短信服务 -->
    < uses - permission android:name = "android.permission.SEND_SMS"/>
⋮
```

运行程序,发送短信界面如图 10-11 所示。



图 10-11　发送短信界面

10.3.2　接收短信

除了从 Android 应用程序发送短信外,还可以在应用程序中使用 BroadcastReceiver 对象接收短信。如果希望应用程序在收到一条特定的短信时执行一个动作,这是很实用的功能,可能根据追踪手机的位置以防丢失或被盗。在这种情况下,可以编写一个应用程序,用来自动侦听包含一些秘密代码的短信。一旦收到此类信息,即可给发送者发回一条包含位置坐标的信息。

下面通过一个案例来演示接收短信。其具体操作步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 Receive_Message。

(2) 打开 res\layout 目录下的 main.xml 布局文件,其代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:orientation = "vertical"
    android:padding = "10dp"
    android:background = "#ffcc66">
    < LinearLayout
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:orientation = "vertical"
        android:gravity = "center_horizontal" >
        < TextView
            android:id = "@ + id/textView1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "短信"
            android:textColor = "#000" />
    </LinearLayout >
</LinearLayout >
```

(3) 打开 src\fs.receive_mesaage 包下的 MainAcitivity.java 文件,在文件中实现短信的接收,其代码为:

```
package fs.receive_message;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends Activity{
    private TextView textView;
    @Override
```

```
        protected void onCreate(Bundle savedInstanceState) {
            //TODO 自动存根法
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            textView = (TextView) findViewById(R.id.textView1);
            Intent intent = getIntent();
            if (intent!= null) {
                String address = intent.getStringExtra("sms_address");
                if (address!= null) {
                    textView.append("\n\n 发件人: \n" + address);
                    String bodyString = intent.getStringExtra("sms_body");
                    if (bodyString!= null) {
                        textView.append("\n 短信内容: \n" + bodyString);
                    }
                }
            }
        }
    }
```

(4) 在 src\fs.receive_mesaage 包下创建一个广播 Broadcast 类,命名为 BroadCastTest.java,其代码为:

```
    package fs.send_message;
    import android.content.BroadcastReceiver;
    import android.content.Context;
    import android.content.Intent;
    import android.os.Bundle;
    import android.telephony.SmsMessage;
    import android.widget.Toast;
    /**
     * 以 BroadcastReceiver 接收短信
     * */
    public class BroadCastTest extends BroadcastReceiver{
        public static final String ACTION = "android.provider.Telephony.SMS_RECEIVED";
        @Override
        public void onReceive(Context context, Intent intent) {
            //TODO 自动存根法
            if (ACTION.equals(intent.getAction())) {
                Intent i = new Intent(context, MainActivity.class);
                i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                SmsMessage[] msgs = getMessageFromIntent(intent);
                StringBuilder sBuilder = new StringBuilder();
                if (msgs!= null && msgs.length > 0) {
                    for (SmsMessage msg : msgs) {
                        sBuilder.append("接收到了短信: \n 发件人是: ");
                        sBuilder.append(msg.getDisplayOriginatingAddress());
                        sBuilder.append("\n------- 短信内容 -------\n");
                        sBuilder.append(msg.getDisplayMessageBody());
                        i.putExtra("sms_address", msg.getDisplayOriginatingAddress());
                        i.putExtra("sms_body", msg.getDisplayMessageBody());
                    }
                }
                Toast.makeText(context, sBuilder.toString(), 1000).show();
                context.startActivity(i);
            }
        }
```

```
        public static SmsMessage[] getMessageFromIntent(Intent intent) {
            SmsMessage retmeMessage[] = null;
            Bundle bundle = intent.getExtras();
            Object pdus[] = (Object[]) bundle.get("pdus");
            retmeMessage = new SmsMessage[pdus.length];
            for (int i = 0; i < pdus.length; i++) {
                byte[] bytedata = (byte[]) pdus[i];
                retmeMessage[i] = SmsMessage.createFromPdu(bytedata);
            }
            return retmeMessage;
        }
    }
```

(5) 打开 AndroidManifest.xml 文件，实现注册广播 Broadcast 及添加短信接收权限，其代码为：

```
    ⋮
            </activity>
            <!-- 注册广播 Broadcast -->
            <receiver android:name = ".BroadCastTest">
        <intent - filter>
            <action android:name = "android.provider.Telephony.SMS_RECEIVED"/>
        </intent - filter>
    </receiver>
        </application>
        <!-- 添加接收短信权限 -->
        <uses - permission android:name = "android.permission.RECEIVE_SMS"/>
    </manifest>
```

运行程序，在系统中发送短信，效果如图 10-12 所示。

10.3.3 群发短信

群发短信是十分实用的功能，逢年过节，很多人都喜欢通过群发短信向自己的朋友表示祝福。群发短信可以将一条短信同时向多个人发送。群发短信的实现十分简单，只要让程序遍历每个收件人的号码并依次向每个收件上发送短信即可。

下面通过一个案例来演示 Android 手机的群发短信。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Group_Message。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明两个 TextView 控件、两个 EditText 控件及两个 Button 控件，其代码为：

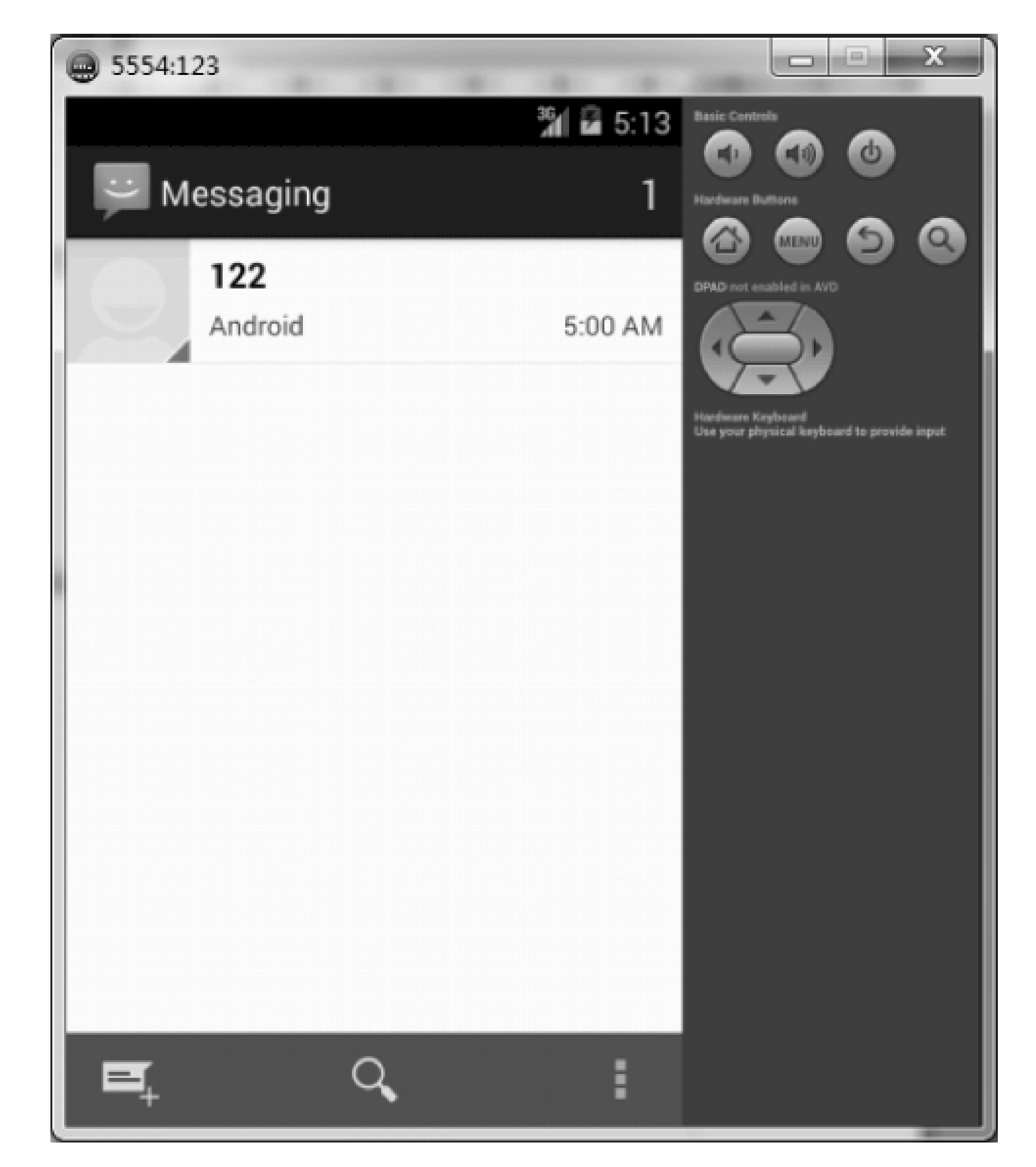


图 10-12 短信接收

```
    <?xml version = "1.0" encoding = "utf - 8"?>
    <LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
```

```
        android:background = "#ffcc66">
    <TextView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "号码(可选多个号码)"/>
    <EditText
        android:id = "@ + id/numbers"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:editable = "false"
        android:cursorVisible = "false"
        android:lines = "2"/>
    <TextView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "短信"/>
    <EditText
        android:id = "@ + id/content"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:lines = "2"/>
    <LinearLayout
        android:orientation = "horizontal"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:gravity = "center_horizontal">
    <Button
        android:id = "@ + id/select"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "选择"/>
    <Button
        android:id = "@ + id/send"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "发送"/>
    </LinearLayout>
    </LinearLayout>
```

(3) 在 res\layout 目录下创建一个 list.xml 文件，用于实现列表选择多个号码，其代码为：

```
<?xml version = "1.0" encoding = "UTF - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
<ListView
    android:id = "@ + id/list"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"/>
</LinearLayout>
```

(4) 打开 src\group_message 包下的 MainActivity.java 文件，在文件中实现短信群发，其代码为：

```
package fs.group_message;
import java.util.ArrayList;
```

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.PendingIntent;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.telephony.gsm.SmsManager;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.Toast;
public class MainActivity extends Activity
{
    EditText numbers, content;
    Button select, send;
    SmsManager sManager;
    //记录需要群发的号码列表
    ArrayList<String> sendList = new ArrayList<String>();
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        sManager = SmsManager.getDefault();
        //获取界面上的文本框、按钮组件
        numbers = (EditText) findViewById(R.id.numbers);
        content = (EditText) findViewById(R.id.content);
        select = (Button) findViewById(R.id.select);
        send = (Button) findViewById(R.id.send);
        //为 send 按钮的单击事件绑定监听器
        send.setOnClickListener(new OnClickListener()
        {
            public void onClick(View v)
            {
                for (String number : sendList)
                {
                    //创建一个 PendingIntent 对象
                    PendingIntent pi = PendingIntent.getActivity(
                        MainActivity.this, 0, new Intent(), 0);
                    //发送短信
                    sManager.sendTextMessage(number, null,
                        content.getText()
                        .toString(), pi, null);
                }
                //提示短信群发完成
```

```
                Toast.makeText(MainActivity.this, "短信群发完成", 8000)
                    .show();
            }
        });
        //为 select 按钮的单击事件绑定监听器
        select.setOnClickListener(new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                //查询联系人的电话号码
                final Cursor cursor = getContentResolver().query(
                    ContactsContract.CommonDataKinds.Phone.CONTENT_URI,null,null, null, null);
                BaseAdapter adapter = new BaseAdapter()
                {
                    @Override
                    public int getCount()
                    {
                        return cursor.getCount();
                    }
                    @Override
                    public Object getItem(int position)
                    {
                        return position;
                    }
                    @Override
                    public long getItemId(int position)
                    {
                        return position;
                    }
                    public View getView1(int position, View convertView,
                        ViewGroup parent)
                    {
                        cursor.moveToPosition(position);
                        CheckBox rb = new CheckBox(MainActivity.this);
                        //获取联系人的电话号码,并去掉中间的横线
                        String number = cursor
                            .getString(
                                cursor
                                 .getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER))
                            .replace("-", "");
                        rb.setText(number);
                        //如果该号码已经被加入发送人名单,默认勾选该号码
                        if (isChecked(number))
                        {
                            rb.setChecked(true);
                        }
                        return rb;
                    }
                    @Override
                    public View getView(int arg0, View arg1, ViewGroup arg2) {
                        //TODO 自动生成的方法存根
                        return null;
                    }
                };
```

```
                //加载 list.xml 布局文件对应的 View
                View selectView = getLayoutInflater().inflate(R.layout.list,null);
                //获取 selectView 中的名为 list 的 ListView 组件
                final ListView listView = (ListView) selectView.findViewById(R.id.list);
                listView.setAdapter(adapter);
                new  AlertDialog.  Builder  ( MainActivity.  this ).  setView  ( selectView ).
setPositiveButton("确定",
                    new DialogInterface.OnClickListener()
                    {
                        @Override
                        public void onClick(DialogInterface dialog,
                            int which)
                        {
                            //清空 sendList 集合
                            sendList.clear();
                            //遍历 listView 组件的每个列表项
                            for (int i = 0; i < listView.getCount(); i++)
                            {
                                CheckBox checkBox = (CheckBox) listView
                                    .getChildAt(i);
                                //如果该列表项被勾选
                                if (checkBox.isChecked())
                                {
                                    //添加该列表项的电话号码
                                    sendList.add(checkBox.getText().toString());
                                }
                            }
                            numbers.setText(sendList.toString());
                        }
                    }).show();
            }
        });
    }
    //判断某个电话号码是否已在群发范围内
    public boolean isChecked(String phone)
    {
        for (String s1 : sendList)
        {
            if (s1.equals(phone))
            {
                return true;
            }
        }
        return false;
    }
}
```

(5) 打开 AndroidManifest.xml 文件，在文件中添加短信群发权限，其代码为：

```
⋮
    </application>
    <!-- 添加群发短信权限 -->
<uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>
<uses-permission android:name="android.permission.SEND_SMS"></uses-permission>
</manifest>
```

运行程序，效果如图 10-13 所示。

程序中提供了一个带列表的对话框供用户选择群发 SMS 的收件人号码，程序则使用了一个 ArrayList<String>集合来保存所有的收件人号码。为了实现群发 SMS 功能，程序使用循环遍历 ArrayList<String>中的每个号码，并使用 SmsManager 依次向每个号码发送短信即可。



图 10-13 短信群发

10.3.4 短信防火墙

每天都会收到各种各样的垃圾短信，短信防火墙是短信软件常见的功能。在 Android 系统中，存在很多系统广播。当手机接收到短信时，通过使用广播的方式来通知所有的应用程序。短信广播是一个有序的广播，一次传递给一个广播接收器，当该接收器处理完成后才会传递给下一个接收器。这样，就可以通过在系统短信程序接收到短信广播之前终止该短信广播，便可实现短信防火墙。

下面通过一个案例来演示短信防火墙功能。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Message_ Firewall。

(2) 对于短信防火墙这类软件是不需要任何界面的。当有短信广播时，接收到广播进行短信判断。在 src\fs. message_ firewall 包下建立一个 SMS_rece. java 文件，其代码为：

```
package fs.message_ firewall;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.util.Log;
import android.widget.Toast;
//接受到短信
public class SMS_rece extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        //TODO 自动存根法
        String action = intent.getAction();
        /* 在 Android 中提供了 SmsMessage 类来管理获取的信息.从短信广播中获取 pdu 数据并转为
SmsMessage 类 */
        if (action.equals(MainActivity.SMS_RECEIVER)) {
            Bundle bundle = intent.getExtras();
            if (bundle!= null) {
                Object[] object = (Object[]) bundle.get("pdus");
                SmsMessage[] messages = new SmsMessage[object.length];
                for (int i = 0; i < object.length; i++) {
                    messages[i] = SmsMessage.createFromPdu((byte[]) object[i]);
                }
                SmsMessage message = messages[0];
                //显示获取短信号码以及短信内容
```

```
                Toast.makeText(
                        context,
                        "接收到消息的号码是: " + message.getDisplayOriginatingAddress()
                                + "\n 接收到的消息是" + message.getMessageBody(), 1000)
                        .show();
                Log.i(MainActivity.TAG, "接收到消息的号码是: "
                    + message.getDisplayOriginatingAddress() + ", 接收到的消息是"
                        + message.getMessageBody());
                /* 通过短信号码进行拦截。如果短信号码为 5566,则禁止该短信广播。这样系统
短信程序将接收不到该广播,不会提示有新的短信,达到短信防火墙的目的 */
                if (message.getDisplayOriginatingAddress().equals("5566")) {
                    abortBroadcast();
                    Log.i(MainActivity.TAG, "终止了短信广播");
                }
            }
        }
    }
}
```

(3) 打开 src\fs.message_ firewall 包下的 MainActivity.java,实现 Activity 类,其代码为:

```
package fs.message_ firewall;
import java.util.ArrayList;
import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.net.Uri;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends Activity {
    static final String TAG = "EXSMS";
    private static final String SENT_SMS_ACTION = "SENT_SMS_ACTION";
    private static final String DELIVERED_SMS_ACTION = "DELIVERED_SMS_ACTION";
    public static final String SMS_RECEIVER = "android.provider.Telephony.SMS_RECEIVED";
    Button btn_sys_send, btn_send;
    EditText in_ph_num, in_sms_text;
    /** 第一次调用活动. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

(4) 打开 AndroidManifest.xml 文件,设置短信防火墙权限,其代码为:

```
⋮
      </activity>
    <!-- 广播注册 -->
      <receiver android:name = ".SMS_receiver" >
```

```
                <intent-filter android:priority="10000">
                    <action android:name="android.provider.Telephony.SMS_RECEIVED" />
                </intent-filter>
            </receiver>
        </application>
        <uses-permission android:name="android.permission.SEND_SMS" />
        <uses-permission android:name="android.permission.RECEIVE_SMS" />
</manifest>
```

10.4 邮件功能

Google 公司在发表 Android 手机平台时，强调的是超强大的网络支持能力，因此，无论通过 GPRS、3G 的电信网络还是 WiFi 的无线 WLAN 网络，都能够发邮件。

发送邮件中使用的 Intent 行为为 android.content.Intent.ACTION_SEND。实际上在 Android 上使用的邮件发送服务是调用 Gmail 程序，而非直接使用 SMTP 协议。

下面通过一个案例来演示在 Android 发送邮件。其具体操作步骤为：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 Send_Email。

（2）打开 res\layout 目录下的 main.xml 文件，在文件中声明两个 EditText 控件、三个 TextView 控件及一个 Button 控件，其代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffcc66">
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="收件人: "
    android:id="@+id/TextView1" />
<EditText
    android:id="@+id/et_to"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"/>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="主题: "
    android:id="@+id/TextView2" />
<EditText
    android:id="@+id/et_subject"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="内容"
    android:id="@+id/TextView3"/>
<EditText
    android:id="@+id/et_content"
    android:layout_width="fill_parent"
```

```
        android:layout_height = "wrap_content"/>
    < Button
        android:id = "@ + id/btn_send"
        android:text = "发送"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
</LinearLayout >
```

(3) 打开 src\fs. send_email 包下的 MainActivity. java 文件,在文件中实现发送邮件功能,其代码为:

```
package fs.send_email;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.view.*;
import android.view.View.OnClickListener;
public class MainActivity extends Activity {
    private EditText et_to, et_subject, et_content;
    private Button btn_send;
    /** 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        et_to = (EditText) findViewById(R.id.et_to);
        et_subject = (EditText) findViewById(R.id.et_subject);
        et_content = (EditText) findViewById(R.id.et_content);
        btn_send = (Button) findViewById(R.id.btn_send);
        //button 的监听器
        btn_send.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                String to = et_to.getText().toString();
                String subject = et_subject.getText().toString();
                String content = et_content.getText().toString();
                //创建 Intent
                Intent emailIntent = new Intent(
                        android.content.Intent.ACTION_SEND);
                //设置内容类型
                emailIntent.setType("plain/text");
                //设置额外信息
        emailIntent.putExtra(android.content.Intent.EXTRA_EMAIL,new String[] { to });
            emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT,subject);
            emailIntent.putExtra(android.content.Intent.EXTRA_TEXT, content);
            //启动 Activity
            startActivity(Intent.createChooser(emailIntent, "发送邮件…"));
            }
        });
    }
}
```

运行程序,效果如图 10-14 所示。



图 10-14　发送邮件

说明:这里是使用 Intent 当作信使,而 Intent 是调用了内置的 ACTION_SEND。此外提供三个 EXTRA_EMAIL 的内置信息,直接使用即可。

```
emailIntent.putExtra(android.content.Intent.EXTRA_EMAIL,new String[] { to });
emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT,subject);
emailIntent.putExtra(android.content.Intent.EXTRA_TEXT, content);
```

10.5　语音识别功能

语音识别在 Android 上使用起来很简单。Android 中主要通过 RecognizerIntent 来实现语音识别,但是如果找不到语音识别设备,就会抛出异常 ActivityNotFoundException,所以需要捕捉这个异常。语音识别在模拟器上是无法测试的,因为语音识别是访问 Google 公司的云端数据,所以如果手机的网络没有开启,就无法实现识别声音的。如果手机不存在语音识别功能的话,也是无法启用识别。

下面通过一个案例来实现 Android 的语音识别。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 Speech_Recognition。

(2) 打开 res\layout 目录下的 main.xml 文件,在文件中声明一个 Button 控件,其代码为:

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:id = "@ + id/LinearLayout1"
    android:layout_width = "match_parent"
    android:layout_height = "wrap_content"
    android:orientation = "vertical"
    tools:context = ".MainActivity"
    android:background = "#ffcc66">
    <Button
```

```
            android:id = "@ + id/btn_speek"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:text = "请说话" />
</LinearLayout>
```

(3) 在 res\layout 目录下创建一个 display_message.xml 文件，在文件中声明一个 ListView 控件，其代码为：

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
        android:layout_width = "match_parent"
        android:layout_height = "match_parent"
        android:orientation = "vertical">
        <ListView
            android:id = "@ + id/list"
            android:layout_width = "match_parent"
            android:layout_height = "0dip"
            android:layout_weight = "1" >
        </ListView>
 </LinearLayout>
```

(4) 打开 src\speech_recognition 包下的 MainActivity.java 文件，在文件中实现语音识别，其代码为：

```
package fs.speech_recognition;
import java.util.ArrayList;
import java.util.List;
import android.app.Activity;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.pm.ResolveInfo;
import android.os.Bundle;
import android.speech.RecognizerIntent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class MainActivity extends Activity implements OnClickListener {
     public final static String EXTRA_MESSAGE = "com.example.androideg.speech.MESSAGE";
     private static final int VOICE_RECOGNITION_REQUEST_CODE = 1001;
     private static final String SPEECH_PROMPT = "请讲话";
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //检查是否存在 recognition activity
        PackageManager pm = getPackageManager();
        List < ResolveInfo > activities = pm.queryIntentActivities(new Intent(
                RecognizerIntent.ACTION_RECOGNIZE_SPEECH), 0);
        Button btn = (Button) findViewById(R.id.btn_speek);
        if (activities.size()!= 0) {
            //如果存在 Activity RECOGNITION 识别,则为按钮绑定点击事件
            btn.setOnClickListener(this);
        } else {
            //如果不存在则禁用按钮
            btn.setEnabled(false);
            btn.setText("语音识别不可用");
```

```
            }
        }
        @Override
        public void onClick(View v) {
            if (v.getId() == R.id.btn_speek){
                startVoiceRecognitionActivity();
            }
        }
        /**
         * 启动语音识别 Activity,接收用户语音输入
         */
        private void startVoiceRecognitionActivity(){
            //通过 Intent 传递语音识别的模式
            Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
            //语言模式: 自由形式的语音识别
             intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_
MODEL_FREE_FORM);
            //提示语音开始
            intent.putExtra(RecognizerIntent.EXTRA_PROMPT,SPEECH_PROMPT);
            //开始执行 Intent、语音识别,并等待返回结果
            startActivityForResult(intent, VOICE_RECOGNITION_REQUEST_CODE);
        }
        @Override
        protected void onActivityResult(int requestCode, int resultCode, Intent data) {
            //确定是语音识别 Activity 返回的结果
            if (requestCode == VOICE_RECOGNITION_REQUEST_CODE){
                //确定返回结果的状态是成功
                if (resultCode == RESULT_OK){
                    //获取语音识别结果
                     ArrayList<String> matches = data.getStringArrayListExtra(RecognizerIntent.
EXTRA_RESULTS);
                    startDisplayMessageActivity(matches);
                }
            }
            super.onActivityResult(requestCode, resultCode, data);
        }
        /**
         * 启动展示 Activity,显示识别结果
         * @param message
         */
        private void startDisplayMessageActivity(ArrayList<String> strList){
            Intent intent = new Intent(this, DisplayMessageActivity.class);
            intent.putExtra(EXTRA_MESSAGE, strList);
            startActivity(intent);
        }
    }
```

(5) 在 src\speech_recognition 包下创建一个 DisplayMessageActivity.java 文件,用于展示识别结果,其代码为:

```
package fs.speech_recognition;
import java.util.ArrayList;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.ArrayAdapter;
```

```
import android.widget.ListView;
public class DisplayMessageActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //设置布局
        setContentView(R.layout.display_message);
        //从 intent 中获取数据
        Intent intent = getIntent();
        ArrayList < String > strList = intent.getStringArrayListExtra(MainActivity.EXTRA_MESSAGE);
        //找到 list,然后赋值
        ListView list = (ListView) findViewById(R.id.list);
        list.setAdapter(new ArrayAdapter < String >(this, android.R.layout.simple_list_item_1,
strList));
    }
}
```

(6) 打开 AndroidManifest.xml 项目配置文件,添加语音识别权限,其代码为:

```
⋮
    </application>
   <!-- 添加权限 -->
< uses - permission android:name = "android.permission.INTERNET" />
</manifest>
```

10.6 多媒体功能

对 Android 手机中多媒体的介绍具体来说就是播放音乐、播放视频、拍照、录制音频视频等知识。

Android 多媒体架构基于第三方 PacketVideo 公司的 OpenCore 来实现,支持所有通用的音频、视频、静态图像格式。Android 平台的音频视频采集、播放的操作都是通过 OpenCore 来实现的,OpenCore 是 Android 多媒体框架的核心。OpenCore 多媒体框架及通用可扩展的接口针对第三方的多媒体编解码器、输入、输出设备等,支持多媒体文件的播放、下载。

10.6.1 音频播放

在 Android 系统中,使用的底层框架库提供了对大部分图像和音视频编码格式的支持,主要包括 MPEG4、H.264、MP3、AAC、AMR、JPG、PNG、GIF 等格式。当然,要完全支持这些格式还需要硬件设备的支持。本小节将介绍在 Android 系统中音频播放的使用。

在多媒体播放中,Android 系统使用了一个名为 MediaPlayer 的类。该类可用来播放音频、视频和流媒体。

下面通过一个案例来实现在 Android 手机中循环播放音乐。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 Player_ Audio。

(2) 打开 res\layout 目录下的 main.xml 文件,在文件中声明 6 个 Button 控件,其代码为:

```
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
```

```
        android:orientation = "vertical"
        android:background = "#ffcc66">
        <Button
            android:id = "@ + id/start"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content" />
        <Button
            android:id = "@ + id/back"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content" />
        <Button
            android:id = "@ + id/next"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content" />
        <Button
            android:id = "@ + id/pause"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content" />
        <Button
            android:id = "@ + id/continuePlay"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content" />
        <Button
            android:id = "@ + id/stop"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content" />
</LinearLayout>
```

（3）在 src\player_ audio 包下创建一个 PlayManager.java 管理类，用来对音频播放进行管理，可以执行上一首、下一首、暂停、继续、播放等的操作，其代码为：

```
package fs.player_audio;
import java.io.File;
import java.util.ArrayList;
import java.util.List;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
public class PlayManager {
    private static PlayManager pManager = null;//单例
 private String[] playStatus = { "end of music list", "top of music list","play" }; //播放状态
    private MediaPlayer mediaPlayer = null;    //播放工具类,系统类
    private String[] fileArray = null;        //播放文件的路径数组,相当于文件列表
    private int playPostition = 0;            //当前播放文件的位置,可用于暂停和继续播放的
    private int currentFilePosition = -1;     //当前播放的是第几个文件
    public static PlayManager getInstance() {
        if (pManager == null) {
            pManager = new PlayManager();
        }
        return pManager;
    }
    private PlayManager() {
//在该管理类初始化时,就读取 MainActivity.URL 下的文件,找出 MP3 文件
/*这里是将路径写在代码中,也可以将它写在一个文件夹浏览处,这样就可以动态的改变路径了*/
        fileArray = getFileArray(MainActivity.URL);
    }
    private String[] getFileArray(String url) {//得到 MainActivity.URL 下的 MP3 文件
```

```
        String[] s = new File(url).list();
        if (s == null || s.length == 0) {          //若 MainActivity.URL 下面没有文件,则直接返回
            return null;
        }
        int length = s.length;
        List<String> list = new ArrayList<String>();   //new 一个集合,只存放 MP3 文件的路径
        for (int i = 0; i < length; i++) {
            if (s[i].endsWith(".MP3")) {
                list.add(s[i]);
            }
        }
        int listSize = list.size();
        if (listSize == 0) {
            return null;
        }
        String[] res = new String[listSize];           //将 MP3 文件路径赋给字符串数组,并将其返回
        for (int i = 0; i < listSize; i++) {
            res[i] = url + list.get(i);
            MainActivity.LogI (res[i]);
        }
        return res;
    }
    public String startPlayVideo() throws Exception {    //开始播放
        currentFilePosition = 0;   /* 因为 currentFilePosition 的初始值为 - 1,所以此处强制赋值
为 0,即播放第一个音频文件,返回播放状态 */
        return playMusic();
    }
    private String playMusic() throws Exception {
    if (currentFilePosition >= fileArray.length) {        //首先判断当前播放的文件是否超过列表
            return playStatus[0];   /* 返回到底了,也可以直接写成 currentFilePosition = 0,这样
就能循环播放列表了 */
        }
        if (currentFilePosition < 0) {
            return playStatus[1];   /* 返回到顶了,也可以写成 currentFilePosition = fileArray
.length,这样反过来循环播放列表 */
        }
        releaseMedia();   /* 每次开始播放列表时,都将 mediaPlayer 释放,这样一边准备下一首或
上一首 */
        mediaPlayer = new MediaPlayer();
        mediaPlayer.setDataSource(fileArray[currentFilePosition]); //设置播放文件的路径
        MainActivity.LogI("this file's path is "
                + fileArray[currentFilePosition]);
        mediaPlayer.prepare();                                   //准备
        mediaPlayer.start();                                     //开始播放
        mediaPlayer.setOnCompletionListener(new OnCompletionListener() {
            public void onCompletion(MediaPlayer mp) {
                try {
                    playNextMusic();            //在每次播放完成之后都用播放下一首
                    MainActivity.LogI("completion this music, play the next");
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
        return playStatus[2];                   //返回正在播放
```

```
        }
        public String playNextMusic() throws Exception {
            currentFilePosition++;              //播放下一首时当前播放文件加 1
            return playMusic();
        }
        public String playBackMusic() throws Exception {
            currentFilePosition--;              //减 1,播放上一首
            return playMusic();
        }
        public void pausePlayMusic() {          //暂停
            if (mediaPlayer!= null) {
                playPosition = mediaPlayer.getCurrentPosition();//得到当前播放的位置
                mediaPlayer.pause();
            }
        }
        public void continuePlay() {                        //继续
            if (mediaPlayer!= null) {
                mediaPlayer.seekTo(playPostition);          //从记录的位置开始播放
                mediaPlayer.start();
            }
        }
        public void stopPlay() {
            if (mediaPlayer!= null) {                       //停止播放器
                mediaPlayer.stop();
            }
        }
        public void releaseMedia() {                        //释放 mediaPlayer 播放对象
            if (mediaPlayer!= null) {
                try {
                    mediaPlayer.release();
                    mediaPlayer = null;
                    System.gc();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    }
```

(4) 打开 src\player_ audio 包下的 MainActivity.java 文件,该文件用于获取管理类,并启动 Activity,其代码为:

```
    package fs.player_audio;
    import android.app.Activity;
    import android.os.Bundle;
    import android.util.Log;
    import android.view.View;
    import android.view.View.OnClickListener;
    import android.widget.Button;
    import android.widget.Toast;
    public class MainActivity extends Activity {
        public static String URL = "mnt/sdcard/";           //音频文件所在路径
        private static String TAG = "zyj";                  //调试
        private static boolean toast = true;
        private static boolean debug = true;
        private static MainActivity mActivity = null;
```

```
private Button start = null;
private Button pause = null;
private Button continuePlay = null;
private Button stop = null;
private Button next = null;
private Button back = null;
private MyListener listener = new MyListener();
private PlayManager pManager = PlayManager.getInstance();//得到管理类实例
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mActivity = this;
    initView();
}
private void initView() {                          //初始化全部的按钮,并设置监听
    start = (Button) findViewById(R.id.start);
    start.setText("开始");
    start.setOnClickListener((OnClickListener) listener);
    pause = (Button) findViewById(R.id.pause);
    pause.setText("暂停");
    pause.setOnClickListener(listener);
    continuePlay = (Button) findViewById(R.id.continuePlay);
    continuePlay.setText("继续播放");
    continuePlay.setOnClickListener(listener);
    stop = (Button) findViewById(R.id.stop);
    stop.setText("停止");
    stop.setOnClickListener(listener);
    next = (Button) findViewById(R.id.next);
    next.setText("下一首");
    next.setOnClickListener(listener);
    back = (Button) findViewById(R.id.back);
    back.setText("上一首");
    back.setOnClickListener(listener);
}
public class MyListener implements OnClickListener {
    public void onClick1(View v) {
        switch (v.getId()) {
        case R.id.start:
            try {
         toastMsg(pManager.startPlayVideo());       //在界面上显示单击之后播放状态
                start.setOnClickListener(null);     //单击开始之后,按钮失效
            } catch (Exception e) {
                e.printStackTrace();
            }
            break;
        case R.id.back:
            try {
                toastMsg(pManager.playBackMusic()); //上一首
            } catch (Exception e) {
                e.printStackTrace();
            }
            break;
        case R.id.next:                             //下一首
            try {
                toastMsg(pManager.playNextMusic());
```

```
                } catch (Exception e) {
                    e.printStackTrace();
                }
                break;
            case R.id.pause:                              //暂停
                pManager.pausePlayMusic();
                break;
            case R.id.continuePlay:                       //继续
                pManager.continuePlay();
                break;
            case R.id.stop:                               //直接停止
                pManager.stopPlay();
                start.setOnClickListener(listener);       //停止之后让开始按钮起作用
                break;
            }
        }
        @Override
        public void onClick(View arg0) {
            //TODO 自动生成的方法存根
        }
    }
    protected void onDestroy() {
        super.onDestroy();
        pManager.releaseMedia();                          //在销毁方法里对播放器进行释放
    }
    public static void LogI(String msg) {
        if (debug) {
            Log.i(TAG, msg);
        }
    }
    public static void toastMsg(String msg) {
        if (toast) {
            Toast.makeText(mActivity, msg, Toast.LENGTH_SHORT).show();
        }
    }
}
```

运行程序,得到如图 10-15 所示。

说明:

加载 MediaPlayer 时,有两种方式。

(1) 第一种方式是利用静态方法 MediaPlayer.create()方式来得到一个 MediaPlayer 实例,这个方法每次都会返回一个 MediaPlayer 实例对象,如果程序需要使用 MediaPlayer 循环播放多个音频文件,此方法就不合适了。

(2) 第二种方式,直接新建一个 MediaPlayer 实例,然后通过 setDataSource()方法来设置文件路径。当调用 setDataSource 方法之后,MediaPlayer 并不会真正地去加载音频文件,而是需要调用 MediaPlayer 中的 prepare 方法准备音频,所谓的"准备",也就是让 MediaPlayer 去加载音频文件。



图 10-15　音频播放界面

10.6.2 音量调节

在实际生活中有时需要根据实际情况设置手机铃声的大小或调整手机的声音模式。在Android平台上提供了能够改变手机声音的解决方案。

本案例是通过getSystemService取得AudioManager对象的，通过该对象的adjustVolume()方法来调整音量的大小。其中，AudioManager.ADJUST_RAISE表示增大音量；AudioManager.ADJUST_LOWER表示减小音量。

getSystemService是Android很重要的一个API，它是Activity的一个方法，根据传入的NAME来取得对应的对象，然后转换成相应的服务对象。

随着音量的增大和减少，手机的声音模式可能会发生变化，这就需要am.getRingerMode()方法获取当前声音模式，并设置相应的图片显示在主界面上。

下面通过一个案例来利用getSystemService获取AudioManager对象实现手机音量的控件。其具体实现步骤为：

(1) 在Eclipse中创建一个Android应用项目，命名为Phone_Volume。

(2) 打开res\layout目录下的main.xml布局文件，在文件中布局两个图片按钮(向下按钮表示减小声音)、用于显示当前音量大小的进度条和用于表示当前声音模式的图片提示；布局两个TextView控件，用于显示提示；布局一个ProgressBar控件，用于控制音量大小；布局一个ImageView控件，用于设置图片按钮的属性，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#ffcc66">
    <LinearLayout
        android:id = "@ + id/LinearLayout01"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:orientation = "horizontal"
        android:gravity = "center">
        <ImageButton
            android:id = "@ + id/ImageView1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:background = "@drawable/up">
        </ImageButton>
        <ImageButton
            android:id = "@ + id/ImageView2"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:background = "@drawable/down">
        </ImageButton>
    </LinearLayout>
    <LinearLayout
        android:id = "@ + id/LinearLayout2"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:orientation = "vertical"
```

```
            android:paddingTop = "10dip">
            <TextView
                android:text = "当前音量大小: "
                android:id = "@ + id/TextView1"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content">
            </TextView>
            <ProgressBar
                android:id = "@ + id/ProgressBar1"
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"
                style = "@android:style/Widget.ProgressBar.Horizontal">
            </ProgressBar>
        </LinearLayout>
        <LinearLayout
            android:id = "@ + id/LinearLayout2"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:orientation = "horizontal"
            android:paddingTop = "10dip">
            <TextView
                android:text = "当前手机状态为: "
                android:id = "@ + id/TextView2"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content">
            </TextView>
            <ImageView
                android:id = "@ + id/ImageView3"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content">
            </ImageView>
        </LinearLayout>
    </LinearLayout>
```

(3) 打开 src\phone_volume 包下的 MainActivity.java 文件,在文件中完成获取手机当前的模式,为 ImageButton 添加监听器,完成对应的功能,其代码为:

```
package fs.phone_volume;
import android.app.Activity;
import android.content.Context;
import android.media.AudioManager;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.ProgressBar;
public class MainActivity extends Activity {
    AudioManager am;
    int volume;
    @Override
    public void onCreate(Bundle savedInstanceState) {
/* 创建 AudioManager 对象,以及创建界面中控件的对象。通过 AudioManager 对象获取当前声音的大小,将获取的值赋予 ProgressBar。根据 AudioManager.getRingerMode()方法获取当前的手机模式,并根据模式设置在主界面上显示图片 */
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.main);
        am = (AudioManager)getSystemService(Context.AUDIO_SERVICE);
        ImageButton up = (ImageButton)this.findViewById(R.id.ImageView1);      //增加声音
    ImageButton down = (ImageButton)this.findViewById(R.id.ImageView2);         //降低声音
    final ProgressBar pb = (ProgressBar)this.findViewById(R.id.ProgressBar1);   //进度条
        final ImageView iv = (ImageView)this.findViewById(R.id.ImageView3);
        volume = am.getStreamVolume(AudioManager.STREAM_RING);          //当前声音大小
        pb.setMax(70);
        pb.setProgress(volume * 10);                                    //设置进度条显示
        int mode = am.getRingerMode();
        if(mode == AudioManager.RINGER_MODE_NORMAL)
        {                                                               //正常模式
            iv.setImageDrawable(getResources().
                    getDrawable(R.drawable.ring));                      //设置图片
        }else if(mode == AudioManager.RINGER_MODE_SILENT)
        {                                                               //静音模式
            iv.setImageDrawable(getResources().
                    getDrawable(R.drawable.jing));                      //设置图片
        }else if(mode == AudioManager.RINGER_MODE_VIBRATE)
        {                                                               //震动模式
            iv.setImageDrawable(getResources().
                    getDrawable(R.drawable.zhen));                      //设置图片
        }
/* 为向上 ImageButton 按钮的监听器,在该监听器中每次单击时调用 am.adjustVolume()方法,并将其
中的第一个参数设置为 AudioManager.ADJUST_RAISE 模式,并根据模式的不同设置不同的提示图片 */
        up.setOnClickListener                                           //声音调大
        (
            new OnClickListener()
            {
                public void onClick(View v) {
                    //TODO 自动存根法
                    am.adjustVolume(AudioManager.ADJUST_RAISE, 0);
                    volume = am.getStreamVolume(AudioManager.STREAM_RING);
                    pb.setProgress(volume * 10);
                    int mode = am.getRingerMode();
                    if(mode == AudioManager.RINGER_MODE_NORMAL)
                    {                                                   //正常模式
                        iv.setImageDrawable(getResources().
                                getDrawable(R.drawable.ring));          //设置图片
                    }else if(mode == AudioManager.RINGER_MODE_SILENT)
                    {                                                   //静音模式
                        iv.setImageDrawable(getResources().
                                getDrawable(R.drawable.jing));          //设置图片
                    }else if(mode == AudioManager.RINGER_MODE_VIBRATE)
                    {                                                   //震动模式
                        iv.setImageDrawable(getResources().
                                getDrawable(R.drawable.zhen));          //设置图片
                    }
                }
            }
        );
/* 为向下 ImageButton 按钮的监听器,在该监听器中每次单击时调用 am.adjustVolume()方法,同样将
其中的第一个参数设置为 AudioManager.ADJUST_LOWER,使手机减少一格音量,将减少后的音量赋值给
ProgressBar,并时刻判断当前手机的模式,根据模式的不同设置不同的提示图片 */
```

```
            down.setOnClickListener                          //声音调小
            (
                new OnClickListener()
                {
                    public void onClick(View v) {
                        //TODO 自动存根法
                        am.adjustVolume(AudioManager.ADJUST_LOWER, 0);
                        volume = am.getStreamVolume(AudioManager.STREAM_RING);
                        pb.setProgress(volume * 10);
                        int mode = am.getRingerMode();
                        if(mode == AudioManager.RINGER_MODE_NORMAL)
                        {                                    //正常模式
                            iv.setImageDrawable(getResources().
                                getDrawable(R.drawable.ring));   //设置图片
                        }else if(mode == AudioManager.RINGER_MODE_SILENT)
                        {                                    //静音模式
                            iv.setImageDrawable(getResources().
                                getDrawable(R.drawable.jing));   //设置图片
                        }else if(mode == AudioManager.RINGER_MODE_VIBRATE)
                        {                                    //震动模式
                            iv.setImageDrawable(getResources().
                                getDrawable(R.drawable.zhen));   //设置图片
                        }
                    }
                }
            );
        }
    }
```

运行程序，默认效果如图 10-16(a)所示。当单击界面中的向上按钮时，即可调大音量，单击界面中的向下按钮时，即可减少音量，当音量减少到 0 时，即将手机调整为震动模式，如图 10-16(b)所示。



(a) 手机为正常模式



(b) 手机为震动模式

图 10-16　调节手机音量

10.6.3 声音录制

在当前使用的智能移动手机中，录音功能也十分常见。在 Android 系统中，MediaRecorder 类用于进行媒体采样，可以完成音频和视频的采集。在 Android 系统中，录音机就是一个典型的使用 MediaRecorder 类的例子。MediaRecorder 类提供了录音、停止等基本功能。可通过 MediaRecorder 类完成录音机音频源、编码格式和输出格式的设置。

在默认情况下，Android 的模拟器不能完成录音的功能。为了能够在模拟器上进行开发调试，需要对模拟器进行设置，使其能够通过计算机的麦克风进行录音。

下面通过一个案例用于实现声音的录制，其实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Voice_Record。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明两个 Button 控件及一个 TextView 控件，其代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#ffcc66">
    <Button
        android:id = "@+id/start"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_alignParentTop = "true"
        android:layout_marginLeft = "18dp"
        android:layout_marginTop = "30dp"
        android:text = "开始" />
    <Button
        android:id = "@+id/stop"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignBaseline = "@+id/start"
        android:layout_alignBottom = "@+id/start"
        android:layout_centerHorizontal = "true"
        android:text = "停止" />
    <TextView
        android:id = "@+id/textView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_above = "@+id/stop"
        android:layout_alignParentLeft = "true"
        android:layout_alignParentTop = "true"
        android:layout_marginBottom = "14dp"
        android:text = "单击按钮开始录制声音" />
</RelativeLayout>
```

(3) 打开 src\fs.voice_record 包下的 MainActivity.java 文件，在文件中实现声明的录制

效果，其代码为：

```
package fs.voice_record;
import java.io.IOException;
import android.app.Activity;
import android.graphics.PixelFormat;
import android.media.MediaRecorder;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
public class MainActivity extends Activity {
    private Button button_start;
    private Button button_stop;
    private MediaRecorder recorder;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getWindow().setFormat(PixelFormat.TRANSLUCENT);          //让界面横屏
        requestWindowFeature(Window.FEATURE_NO_TITLE);           //去掉界面标题
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        //重新设置界面大小
        setContentView(R.layout.main);
        init();
    }
    private void init() {
        button_start = (Button) this.findViewById(R.id.start);
        button_stop = (Button) this.findViewById(R.id.stop);
        button_stop.setOnClickListener(new AudioListerner());
        button_start.setOnClickListener(new AudioListerner());
    }
    class AudioListerner implements OnClickListener {
        @Override
        public void onClick(View v) {
            if (v == button_start) {
                initializeAudio();
            }
            if (v == button_stop) {
                recorder.stop();              //停止刻录
                recorder.release();           //刻录完成一定要释放资源
            }
        }
        private void initializeAudio() {
            recorder = new MediaRecorder();   //新建一个 MediaRecorder 对象
            recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
            //设置 MediaRecorder 的音频源为麦克风
            recorder.setOutputFormat(MediaRecorder.OutputFormat.RAW_AMR);
            //设置 MediaRecorder 录制的音频格式
            recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
            //设置 MediaRecorder 录制音频的编码为 amr
            recorder.setOutputFile("/sdcard/peipei.amr");
            //设置录制的音频文件保存路径
            try {
                recorder.prepare();           //准备录制
```

```
                recorder.start();                //开始录制
            } catch (IllegalStateException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

(4) 打开 AndroidManifest.xml 配置文件,为文件添加相应权限,其代码为:

```
⋮
        </activity>
    </application>
  <!-- 联网权限 -->
  <uses-permission android:name="android.permission.INTERNET" />
  <!-- 往 SDCard 写入数据权限 -->
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
  <!-- 录音权限 -->
  <uses-permission android:name="android.permission.RECORD_AUDIO" />
  <!-- 在 SDCard 中创建与删除文件权限 -->
  <uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
</manifest>
```

运行程序,效果如图 10-17 所示。



图 10-17 声音录制

10.6.4 视频播放

视频在人们生活和娱乐中是不可缺少的,Android 为视频播放提供了 VideoView 和 MediaController 两个现成的组件,可以方便地实现 MP4、3GP 等视频的播放。

注意:如果播放时无法播放或只有声音没有图像,就需要换压缩软件和调整压缩参数重新压缩视频了,暂时只能这样。这一点是非常重要的,如果没有压缩好,就不会看见视频的效果。

下面通过一个案例来演示视频播放效果。其实现具体步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 Video_View。

(2) 打开 res\layout 目录下的 main.xml 布局文件,在文件中声明一个 SurfaceView 控件、一个 TextView 控件、一个 SeekBar 控件、三个 Button 控件及一个 MediaController 控件,其代码为:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffcc66">
```

```
    <SurfaceView
      android:id="@+id/VideoView01"
      android:layout_width="320dip"
      android:layout_height="240dip">
    </SurfaceView>
    <TextView
      android:id="@+id/TextView01"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textColor="#000000"
      android:textSize="20dip"
      android:text="0m:0s">
    </TextView>
    <SeekBar
      android:id="@+id/SeekBar01"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content">
    </SeekBar>
    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="播放" />
    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="暂停" />
    <Button
        android:id="@+id/button3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="停止" />
    <MediaController
      android:id="@+id/MediaController01"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"/>
    <LinearLayout
      android:id="@+id/LinearLayout01"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:orientation="horizontal">
    </LinearLayout>
</LinearLayout>
```

（3）打开 src\fs.video_view 包下的 MainActivity.java 文件，在文件中实现视频的播放，其代码为：

```
package fs.video_view;
import android.app.Activity;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
```

```
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.SeekBar.OnSeekBarChangeListener;
public class MainActivity extends Activity
implements OnClickListener,OnSeekBarChangeListener
{
    public static final int UPDATE_TIME = 0;                    //更新播放时间的消息编号
    public static final String currentPlay = "/sdcard/ * .3gp";//播放路径
    Button play;                                                //播放按钮
    Button pause;                                               //暂停按钮
    Button stop;                                                //停止按钮
    SurfaceView sv;                                             //播放显示用的 SurfaceView
    SurfaceHolder sh;                                           //播放用的 SurfaceHolder
    MediaPlayer mp;                                             //媒体播放器
    SeekBar sb;                                                 //进度显示拖拉条
    TextView tvTime;                                            //时间长度显示
    Handler hd;                                                 //消息处理器
    int state = 0;              //播放状态指示; 0 表示未准备; 1 表示播放中; 2 表示暂停中
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //为播放、暂停、停止三个按钮添加监听器
        play = (Button)this.findViewById(R.id.button1);
        pause = (Button)this.findViewById(R.id.button2);
        stop = (Button)this.findViewById(R.id.button3);
        play.setOnClickListener(this);
        pause.setOnClickListener(this);
        stop.setOnClickListener(this);
        //初始化播放用的 SurfaceView 及 SurfaceHolder
        sv = (SurfaceView)this.findViewById(R.id.VideoView01);
        sh = sv.getHolder();
        sh.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
        //初始化进度拖拉条引用
        sb = (SeekBar)this.findViewById(R.id.SeekBar01);
        //不在暂停状态禁用拖拉条
        sb.setEnabled(false);
        //给进度拖拉条添加监听器
        sb.setOnSeekBarChangeListener(this);
        //初始化显示播放时长的文本框
        tvTime = (TextView)findViewById(R.id.TextView01);
        //线程中创建一个 Handler
        hd = new Handler()
        {
            @Override
            public void handleMessage(Message msg)
            {
                //调用父类处理
```

```
                    super.handleMessage(msg);
                    //根据消息 what 编号的不同,执行不同的业务逻辑
                    switch(msg.what)
                    {
                       //将消息中的内容提取出来显示在 Toast 中
                       case UPDATE_TIME:
                           //获取消息中的数据
                           Bundle b = msg.getData();
                           //获取内容字符串
                           String msgStr = b.getString("msg");
                           //设置字符串到显示录音时长的文本框中
                           tvTime.setText(msgStr);
                       break;
                    }
                }
            };
        }
        public void onClick(View v) {
            if(v == play)
            {//按下播放按钮
               if(state == 1)
               {//若当前已经是播放状态,则报错返回
                   Toast.makeText
                   (
                       this,
                       "播放中,请结束本次播放再开始新的播放!",
                       Toast.LENGTH_SHORT
                   ).show();
                   return;
               }
               else if(state == 0)
               {//若当前是未准备状态,则进行播放前的准备工作
                   //创建媒体播放器对象
                   mp = new MediaPlayer();
                   //设置音频流格式
                   mp.setAudioStreamType(AudioManager.STREAM_MUSIC);
                   //设置播放显示用 SurfaceView 的 SurfaceHolder
                   mp.setDisplay(sh);
                   //给视频播放结束事件添加的监听器
                   mp.setOnCompletionListener(
                         new OnCompletionListener()
                            {
                                public void onCompletion(MediaPlayer arg0)
                                {//歌曲播放结束停止播放并更新界面状态
                                    state = 2;
                                }
                            }
                   );
                   try
                   {
                       //设置播放路径
                       mp.setDataSource(currentPlay);
                       //准备播放
                       mp.prepare();
                   }
                   catch(Exception e)
```

```
            {
                e.printStackTrace();
            }
            //根据片长设置拖拉条最大值
            sb.setMax(mp.getDuration()/1000);
        }
        mp.start();                          //开始播放
        state = 1;                           //状态设置为 1,表示播放中
        sb.setEnabled(false);                //禁用拖拉条
        new Thread()                         //启动一个线程定时更新进度条及时间
        {
            public void run()
            {
                while(state == 1)
                {
                    sb.setProgress(mp.getCurrentPosition()/1000);
                    setTime(mp.getCurrentPosition()/1000);
                    try
                    {
                        Thread.sleep(1000);
                    }
                    catch(Exception e)
                    {
                        e.printStackTrace();
                    }
                }
            }
        }.start();
    }
    else if(v == pause)
    {                                        //按下暂停按钮
        if(state!= 1)
        {                                    //若当前不是播放状态
            Toast.makeText
            (
                this,
                "请在播放状态再暂停!",
                Toast.LENGTH_SHORT
            ).show();
            return;
        }
        mp.pause();                          //暂停
        state = 2;                           //设置状态为 2,表示暂停
        sb.setEnabled(true);                 //启用拖拉条
    }
    else if(v == stop)
    {//按下停止按钮
        if(state == 0)
        {
            return;
        }
        state = 0;
        mp.stop();                           //停止播放
        mp.release();                        //释放播放器
        mp = null;                           //清空引用
        sb.setEnabled(false);                //禁用拖拉条
        sb.setProgress(0);                   //设置进度为 0
        setTime(0);                          //设置时间为 0
```

```
            }
        }
         //设置显示时间的方法
        public void setTime(int countSecond)
        {
            //计算分钟和秒
            int second = countSecond % 60;
            int minute = countSecond/60;
            //创建内容字符串
            String msgStr = minute + "m:" + second + "s";
            //创建消息数据 Bundle
            Bundle b = new Bundle();
            //将内容字符串放进数据 Bundle 中
            b.putString("msg", msgStr);
            //创建消息对象
            Message msg = new Message();
            //设置数据 Bundle 到消息中
            msg.setData(b);
            //设置消息的 what 值
            msg.what = UPDATE_TIME;
            //发送消息
            hd.sendMessage(msg);
        }
        //实现进度拖拉的监听方法
        public void onProgressChanged(SeekBar seekBar, int progress,
                boolean fromUser) {
            if(mp!= null&&state == 2)
            {
                //进度拖拉到指定位置
                mp.seekTo(progress * mp.getDuration()/sb.getMax());
            }
        }
        public void onStartTrackingTouch(SeekBar seekBar) { }
        public void onStopTrackingTouch(SeekBar seekBar) { }
    }
```

运行程序,效果如图 10-18 所示。



图 10-18　视频播放

10.6.5　照相机实现

手机一般都会提供相机功能,有些相机的镜头甚至支持 800 万以上像素,有些甚至支持光学变焦,这些手机已经变成了专业数字相机。

Camera 类位于 android.hardware 软件包,其是 Android 照相机拍照操作的主要类。其提供了多种函数,包括打开相机、预览、设置相机参数和关闭相机等。通过这些方法应用程序可很方便地实现照相机的拍照操作。

Android 应用 Camera 来控制拍照比较简单,其实现步骤为:

(1) 调用 Camera 的 open()方法打开相机。

(2) 调用 Camera 的 getParameters()方法获

取拍照参数。该方法返回一个 Camera. Parameters 对象。

(3) 调用 Camera. Parameters 对象方法设置拍照参数。

(4) 调用 Camera 的 setParameters()方法,并将 Camera. Parameters 对象作为参数传入,这样即可对相机的拍照参数进行控制。

(5) 调用 Camera 的 startPreview()方法开始预览取景,在预览取景前需要调用 Camera 的 setPreviewDisplay(SurfaceHolder holder)方法设置使用哪个 SurfaceView 来显示取景图片。

(6) 调用 Camera 的 takePicture()方法进行拍照。

(7) 结束程序时,调用 Camera 的 stopPreview()方法结束取景预览,并调用 release()方法释放资源。

下面通过一个实例来实现调用系统相机拍照,将其显示在屏幕上,并且将照片存到 SD 卡。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 Camera_View。

(2) 打开 res\layout 目录下的 main. xml 布局文件,在文件中声明一个 ImageView 控件、一个 SurfaceView 控件以及两个 Button 控件,其代码为:

```
<FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:id = "@ + id/FrameLayout1"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent" >
    <!-- 显示预览图形 -->
    <SurfaceView
        android:id = "@ + id/surfaceView"
        android:layout_width = "match_parent"
        android:layout_height = "match_parent"/>
    <!-- 相对布局,放置两个按钮 -->
        <RelativeLayout
            android:id = "@ + id/buttonLayout"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:visibility = "gone">
        <!-- 拍照按钮 -->
        <Button
            android:id = "@ + id/takepicture"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_alignParentRight = "true"
            android:layout_alignParentBottom = "true"
            android:background = "#AABBCC"
            android:onClick = "btnOnclick"/>
        <ImageView
            android:id = "@ + id/scalePic"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_alignParentLeft = "true"
            android:layout_alignParentBottom = "true"
            android:layout_marginLeft = "5dp"
            android:background = "#AABBCC"
            android:onClick = "imageClick"/>
```

```
    </RelativeLayout>
</FrameLayout>
```

（3）打开 src\camera_view 包下的 MainActivity.java 文件，用于实现相机的拍摄，其代码为：

```
package fs.camera_view;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import android.app.Activity;
import android.content.Intent;
import android.graphics.PixelFormat;
import android.hardware.Camera;
import android.hardware.Camera.PictureCallback;
import android.os.Bundle;
import android.os.Environment;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.Surface;
import android.view.SurfaceHolder;
import android.view.SurfaceHolder.Callback;
import android.view.SurfaceView;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;
/**
 * Android 手机拍照
 */
public class MainActivity extends Activity {
    private View layout;
    private Camera camera;
    private Camera.Parameters parameters = null;
    Bundle bundle = null;                          //声明一个 Bundle 对象，用来存储数据
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //显示界面
        setContentView(R.layout.main);
        layout = this.findViewById(R.id.buttonLayout);
        SurfaceView surfaceView = (SurfaceView) this
                .findViewById(R.id.surfaceView);
        surfaceView.getHolder()
                .setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
        surfaceView.getHolder().setFixedSize(176, 144);  //设置 Surface 分辨率
        surfaceView.getHolder().setKeepScreenOn(true);   //屏幕常亮
        surfaceView.getHolder().addCallback(new SurfaceCallback());//为 SurfaceView 的句柄
                                                         //添加一个回调函数
    }
    /**
     * 按钮被单击触发的事件
     */
    public void btnOnclick(View v) {
        if (camera!= null) {
```

```
            switch (v.getId()) {
            case R.id.takepicture:
                //拍照
                camera.takePicture(null, null, new MyPictureCallback());
                break;
            }
        }
    }
    /**
     * 图片被单击触发的时间
     */
    public void imageClick(View v) {
        if (v.getId() == R.id.scalePic) {
            if (bundle == null) {
                 Toast.makeText(getApplicationContext(), R.string.takephoto, Toast.LENGTH_
SHORT).show();
            } else {
                Intent intent = new Intent(this, ShowPicActivity.class);
                intent.putExtras(bundle);
                startActivity(intent);
            }
        }
    }
    private final class MyPictureCallback implements PictureCallback {
        @Override
        public void onPictureTaken(byte[] data, Camera camera) {
            try {
                bundle = new Bundle();  bundle.putByteArray("bytes", data); //将图片字节数
//据保存在 bundle 当中,实现数据交换
                saveToSDCard(data);                         //保存图片到 SD 卡中
                  Toast.makeText(getApplicationContext(), R.string.success, Toast.LENGTH_
SHORT).show();
                camera.startPreview();                      //拍完照后,重新开始预览
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
    /**
     * 将拍下来的照片存放在 SD 卡中
     */
    public static void saveToSDCard(byte[] data) throws IOException {
        Date date = new Date();
        SimpleDateFormat format = new SimpleDateFormat("yyyyMMddHHmmss"); //格式化时间
        String filename = format.format(date) + ".jpg";
        File fileFolder = new File(Environment.getExternalStorageDirectory()
                + "/finger/");
        if (!fileFolder.exists()) {            //如果目录不存在,则创建一个名为 finger 的目录
            fileFolder.mkdir();
        }
        File jpgFile = new File(fileFolder, filename);
        FileOutputStream outputStream = new FileOutputStream(jpgFile); //文件输出流
        outputStream.write(data);              //写入 SD 卡中
        outputStream.close();                  //关闭输出流
    }
```

```
private final class SurfaceCallback implements Callback {
    //拍照状态变化时调用该方法
    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width,
            int height) {
        parameters = camera.getParameters();            //获取各项参数
        parameters.setPictureFormat(PixelFormat.JPEG);  //设置图片格式
        parameters.setPreviewSize(width, height);       //设置预览大小
        parameters.setPreviewFrameRate(5);              //设置每秒显示 4 帧
        parameters.setPictureSize(width, height);       //设置保存的图片尺寸
        parameters.setJpegQuality(80);                  //设置照片质量
    }
    //开始拍照时调用该方法
    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        try {
            camera = Camera.open();                     //打开摄像头
        camera.setPreviewDisplay(holder);     //设置用于显示拍照影像的 SurfaceHolder 对象
            camera.setDisplayOrientation(getPreviewDegree(MainActivity.this));
            camera.startPreview();                      //开始预览
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    //停止拍照时调用该方法
    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
        if (camera!= null) {
            camera.release();                           //释放照相机
            camera = null;
        }
    }
}
/**
 * 单击手机屏幕后,显示两个按钮
 */
@Override
public boolean onTouchEvent(MotionEvent event) {
    switch (event.getAction()) {
    case MotionEvent.ACTION_DOWN:
        layout.setVisibility(ViewGroup.VISIBLE);        //设置视图可见
        break;
    }
    return true;
}
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    switch (keyCode) {
    case KeyEvent.KEYCODE_CAMERA:                       //按下拍照按钮
        if (camera!= null && event.getRepeatCount() == 0) {
            /* 调用 takePicture()方法进行拍照是传入了一个 PictureCallback 对象——当程
序获取了拍照所得的图片数据之后,PictureCallback 对象将会被回调,该对象可以负责对相片进行保
存或传入网络 */
            camera.takePicture(null, null, new MyPictureCallback());
        }
```

```
        }
        return super.onKeyDown(keyCode, event);
    }
    //提供一个静态方法,用于根据手机方向获得相机预览画面旋转的角度
    public static int getPreviewDegree(Activity activity) {
        //获得手机的方向
        int rotation = activity.getWindowManager().getDefaultDisplay()
                .getRotation();
        int degree = 0;
        //根据手机的方向计算相机预览画面应该选择的角度
        switch (rotation) {
        case Surface.ROTATION_0:
            degree = 90;
            break;
        case Surface.ROTATION_90:
            degree = 0;
            break;
        case Surface.ROTATION_180:
            degree = 270;
            break;
        case Surface.ROTATION_270:
            degree = 180;
            break;
        }
        return degree;
    }
}
```

(4) 在 src\camera_view 包下创建一个 ShowPicActivity.java 文件,实现图片的显示,其代码为:

```
package fs.camera_view;
import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.os.Bundle;
import android.widget.ImageView;
public class ShowPicActivity extends Activity {
    private ImageView ivPic = null;                          //显示图片控件
    /**
     * Activity 在创建时回调函数,用来初始化变量
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ivPic = (ImageView) findViewById(R.id.scalePic);
        setImageBitmap(getImageFormBundle());

    }
    /**
     * 将 MainActivity 传过来的图片显示在界面当中
     */
    public void setImageBitmap(byte[] bytes) {
```

```
            Bitmap cameraBitmap = byte2Bitmap();
            //根据拍摄的方向旋转图像(纵向拍摄时要需要将图像选择 90°)
            Matrix matrix = new Matrix();
            matrix.setRotate(MainActivity.getPreviewDegree(this));
            cameraBitmap = Bitmap
                    .createBitmap(cameraBitmap, 0, 0, cameraBitmap.getWidth(),
                            cameraBitmap.getHeight(), matrix, true);
            ivPic.setImageBitmap(cameraBitmap);
        }
        /*
         * 从 Bundle 对象中获取数据
         */
        public byte[] getImageFormBundle() {
            Intent intent = getIntent();
            Bundle data = intent.getExtras();
            byte[] bytes = data.getByteArray("bytes");
            return bytes;
        }
        /**
         * 将字节数组的图形数据转换为点阵图
         */
        private Bitmap byte2Bitmap() {
            byte[] data = getImageFormBundle();
            //将 byte 数组转换成 Bitmap 对象
            Bitmap bitmap = BitmapFactory.decodeByteArray(data, 0, data.length);
            return bitmap;
        }
}
```

(5) 打开 res\values 包下的 strings.xml 文件，实现变量声明，其代码为：

```
<?xml version = "1.0" encoding = "utf-8"?>
<resources>
    <string name = "app_name">照相机</string>
    <string name = "action_settings">Settings</string>
    <string name = "hello_world">Hello world!</string>
    <string name = "takephoto">拍一张</string>
    <string name = "success">成功</string>
</resources>
```

(6) 打开 AndroidManifest.xml 文件，实现照片机权限设置，其代码为：

```
⋮
    <uses-sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
  <uses-permission android:name = "android.permission.CAMERA" />
    <!-- 在 SD 卡中创建与删除文件权限 -->
    <uses-permission android:name = "android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
    <!-- 往 SD 卡写入数据权限 -->
    <uses-permission android:name = "android.permission.WRITE_EXTERNAL_STORAGE" />
```

10.6.6 语音朗读

Android 系统能够将用户输入的文本内容朗读出来。Android 系统的文本语音朗读功能通过 TextToSpeech 来实现。目前，Android 系统提供了英语、法语、德语、意大利语和西班牙

语的文本语音转换功能，但暂时还不支持中文朗读功能。其中，英语还可以选择美式发音或英式发音。通过 Android 语音朗读可以实现很多有趣的应用，如短信自动阅读、电子书朗读软件等。

在 Android 中使用语音朗读功能只需要使用 TextToSpeech 类，该类实现了很多关于语音的功能，使用该类必须为其设置语言，支持语言列表位于 java. util 类里的 Local 类，具体如表 10-3 所示。

表 10-3　Local 类国家及语言代码

国家/语言代码	含　义	国家/语言代码	含　义
Locale. CANADA	加拿大英语	Locale. ITALIAN	意大利语
Locale. CANADA_FRENCH	加拿大法语	Locale. ITALY	意大利
Locale. CHINA	中国	Locale. JAPAN	日本
Locale. CHINESE	中文	Locale. JAPANESE	日语
Locale. ENGLISH	英国	Locale. KOREA	韩国
Locale. FRANCE	法国	Locale. KOREAN	韩语
Locale. FRENCH	法语	Locale. SIMPLIFIED_CHINESE	简体中文
Locale. GERMAN	德语	Locale. UK	英式英语
Locale. GERMANY	德国	Locale. US	美式英语

下面通过一个案例来实现 Android 手机的语音朗读。其具体实现步骤为：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 Localizer_Test。

（2）打开 res\layout 目录下的 main. xml 布局文件，在文件中声明一个 EditText 控件、两个 Button 控件及一个 CheckBox 控件，其代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
 < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
 android:layout_width = "fill_parent"
 android:layout_height = "fill_parent"
 android:orientation = "vertical"
 android:background = "＃fff666">
 < EditText
 android:layout_width = "fill_parent"
 android:layout_height = "wrap_content"
 android:id = "@ + id/edittext"/>
 < Button
 android:layout_width = "wrap_content"
 android:layout_height = "wrap_content"
 android:text = "读"
 android:id = "@ + id/rbutton"/>
 < Button
 android:layout_width = "wrap_content"
  android:layout_height = "wrap_content"
  android:text = "保存"
  android:id = "@ + id/sbutton"/>
  < CheckBox
  android:id = "@ + id/checkbox"
  android:layout_width = "fill_parent"
  android:layout_height = "wrap_content"
  android:text = "边写边读"
 android:checked = "true" />
 </LinearLayout >
```

(3) 打开 src\localizer_test 包下的 MainActivity.java 文件,在文本将内容转换成语音并朗读出来:可以一次全部朗读出来,也可以边写边读;可以将文本保存为语音文件,其代码为:

```
package fs.localizer_test;
import java.util.Locale;
import android.app.Activity;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends Activity
{
    private EditText mEditText = null;
    private Button readButton = null;
    private Button saveButton = null;
    private CheckBox mCheckBox = null;
    private TextToSpeech mTextToSpeech = null;
    /** 第一次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mEditText = (EditText)this.findViewById(R.id.edittext);
        readButton = (Button)this.findViewById(R.id.rbutton);
        saveButton = (Button)this.findViewById(R.id.sbutton);
        mCheckBox = (CheckBox)this.findViewById(R.id.checkbox);
        //实例并初始化 TTS 对象
        mTextToSpeech = new TextToSpeech(this,new TextToSpeech.OnInitListener()
        {
            @Override
            public void onInit(int status)
            {
                //TODO 自动存根法
                if(status == TextToSpeech.SUCCESS)
                {
                    int supported = mTextToSpeech.setLanguage(Locale.US);
                    mTextToSpeech.setSpeechRate(1);
                    if((supported!= TextToSpeech.LANG_AVAILABLE)&&(supported!= TextToSpeech.LANG_
COUNTRY_AVAILABLE))
                    {
                        displayToast("不支持当前语言!");
                    }
                }
            }
        });
        //朗读按钮监听
        readButton.setOnClickListener(new View.OnClickListener()
        {
            @Override
```

```
            public void onClick(View v)
            {
                //TODO 自动存根法
                //朗读 EditText 里的内容
                mTextToSpeech.speak(mEditText.getText().toString(), TextToSpeech.QUEUE_
FLUSH, null);
            }
        });
        //保存按钮监听
        saveButton.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                //TODO 自动存根法
                //将 EditText 里的内容保存为语音文件
                int r = mTextToSpeech.synthesizeToFile(mEditText.getText().toString(), null,
"/mnt/sdcard/speak.wav");
                if(r == TextToSpeech.SUCCESS)
                    displayToast("保存成功!");
            }
        });
        //EditText 内容变化监听
        mEditText.addTextChangedListener(mTextWatcher);
    }
    private TextWatcher mTextWatcher = new TextWatcher()
    {
        @Override
        public void afterTextChanged(Editable s)
        {
            //TODO 自动存根法
            //如果是边写边读
            if(mCheckBox.isChecked()&&(s.length()!= 0))
            {
                //获得 EditText 的所有内容
                String t = s.toString();
                mTextToSpeech.speak(t.substring(s.length() - 1), TextToSpeech.QUEUE_FLUSH, null);
            }
        }
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after)
        {
            //TODO 自动存根法
        }
        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count)
        {
            //TODO 自动存根法
        }
    };
    //显示 Toast 函数
    private void displayToast(String s)
    {
        Toast.makeText(MainActivity.this, s, Toast.LENGTH_SHORT).show();
    }
```

```
        @Override
        public void onDestroy()
        {
            super.onDestroy();
            if(mTextToSpeech!= null)
                mTextToSpeech.shutdown();                         //关闭 TTS
        }
}
```

(4) 打开 AndroidManifest.xml 文件,设置语音朗读权限,其代码为:

```
⋮
    </application>
    <!-- 语音权限 -->
<uses-permission android:name = "android.permission.WRITE_EXTERNAL_STORAGE"/>
</manifest>
```

运行程序,在编辑框中输入相应的字串,即实现边读边保存功能,效果如图 10-19 所示。



图 10-19　语音朗读

10.7　Android 定位功能

Android 支持 GPS 和网络地图,通常将各种不同的定位技术称为 LBS。LBS 是基于位置的服务(Location Based Service)的简称,是通过电信移动运营商的无线电通信网络(如 GSM 网、CDMA 网)或外部定位方式(如 GPS)获取移动终端用户的位置信息(地理坐标或大地坐标),在地理信息系统(Geographic Information System,GIS)平台的支持下,为用户提供相应服务的一种增值业务。

10.7.1　GPS 概述

在实现 GPS 定位前,先了解一下 GPS 的部分特性:

(1) GPS 定位需要依靠三颗或三颗以上的卫星。

(2) GPS 定位受环境影响较大,在晴朗的空地上,较容易搜索到卫星,而在室内通常是无法搜索到卫星的。

(3) GPS 定位需要使用 GPS 功能模块,而 GPS 功能模块的耗电量是巨大的。

在 Android 系统中,实现 GPS 定位的思路应该是:

(1) 获取 GPS 的 Location Provider。

(2) 将此 Provider 传入到 requestLocationUpdates()方法,让 Android 系统获知搜索位置方式。

(3) 实现了 GpsStatus. Listener 接口的对象,重写 onGpsStatusChanged()方法,向 LocationManager 添加此监听器,检测卫星状态(可选步骤)。

10.7.2 GPS 状态

在 Android 中提供了对应方法用于检查 GPS 的状态。

1. 获取首次定位时间

在 Android 中提供了 getTimeToFirstFix()方法用于获取 GPS 的首次定位时间。getTimeToFirstFix()方法用于获取最新的 GPS 引擎重新启动以致收到的首次定位所需的时间,其单位为毫秒(ms)。

注意: *在空旷的地方,GPS 定位时间较短;而障碍物较多的地方,定位时间较长。*

getTimeToFirstFix()方法的调用格式为:

```
public int getTimeToFirstFix()
```

2. 获取最大卫星数量

在 Android 中提供了 getMaxStatellites()方法用于获取在卫星列表中可返回的最大卫星数目。这个数目是 getStatellites()方法能够得到的最大卫星数目,但并不代表当前实际获得的卫星数量。其一般返回值为 255。

getMaxStatellites 方法的调用格式为:

```
public int getMaxStatellites()
```

3. 获取 GPS 卫星状态

在 Android 中了 getStatellites()方法用于获取 GPS 引擎的当前状态,该方法返回一个为 GpsStatellites()的对象数组值,其中包含了卫星列表。

getStatellites 方法的调用格式为:

```
public Iterable<GpsSatellite> getSatellites()
```

10.7.3 GPS 位置信息

在 Android 中提供了相关函数用于获取 GPS 位置信息,包括精度、方位、经纬度、海拔、速度、高度、运营商收费等信息。

1. 精度

在 Android 中提供了 getAccuracy()方法用于获取当前定位数据的精确度信息,其返回值为 float 类型的数据,单位为米(m)。精度反映了经度与纬度在定位上的误差。getAccuracy()的调用方法为:

```
Public float getAccuracy()
```

2. 方位

在 Android 中提供了 getBearing()方法用于获取 GPS 卫星的方位,其返回值为 float 类型。方位以地理北为参考,取值范围为 0～360°。getBearing()的调用方法为:

```
Public float getBearing()
```

3. 高度

在 Android 中提供了 getAltitude()方法用于获取 GPS 卫星的高度,其返回值为 float 类型。getAltitude()的调用方法为:

```
Public float getAltitude()
```

4. 经度

在 Android 中提供了 getLatitude()方法用于获取 GPS 卫星的经度,其返回值为 float。getLatitude()的调用方法为:

```
public float getLatitude()
```

5. 海拔

在 Android 中提供了 getAltitude()方法用于获取 GPS 卫星的海拔,其返回值为 float。getAltitude()的调用方法为:

```
public float getAltitude()
```

6. 纬度

在 Android 中提供了 getLongitude()方法用于获取 GPS 卫星的纬度,其返回值为 float。getLongitude()的调用方法为:

```
Public float getLongitude()
```

7. 速度

在 Android 中提供了 getSpeed()方法用于获取 GPS 卫星的速度,其返回值为 float。getSpeed()的调用方法为:

```
Public float getSpeed()
```

10.7.4 GPS 参数

在 Android 中提供了相关函数用于获取 GPS 参数,包括方位角、高度角、伪随机数、信噪比等信息。

1. 方位角

在 Android 中提供了 getAzimuth()方法用于获取方位角,其返回值为 true,方位角的取值范围为 0～360°。getAzimuth()调用方法为:

```
public float getAzimuth()
```

2. 高度角

在 Android 中提供了 getElevation()方法用于获取 GPS 卫星的高度角,其返回值为 true,高度角的取值范围为 0～360°。getElevation 调用方法为:

```
public float getElevation()
```

3. 伪随机数

在 Android 中提供了 getPrn()方法用于获取 GPS 卫星的伪随机数(PRN),其返回值为

int 类型。getPrn()的调用方法为：

```
public int getPrn()
```

4. 信噪比

在 Android 中提供了 getSnr()方法用于获取 GPS 卫星的信噪比，其返回值为 float。getSnr()的调用方法为：

```
Public float getSnr()
```

10.7.5 GPS 实用经典案例

前面介绍了 GPS 概念及 Android 开发 GPS 应用涉及的常用类和方法。在本小节中，开发一个小应用，实时获取定位信息，包括用户所在的纬度、经度、高度、方向、移动速度等。

其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 GPS_Test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 EditText 控件，用于显示 GPS 的位置信息，其代码为：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:background="#fff666">
    <EditText
        android:id="@+id/main_et_show"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:cursorVisible="false"
        android:editable="false" />
</RelativeLayout>
```

(3) 打开 src\fs.gps_test 包下的 MainActivity.java 文件，在文件中实现 GPS 动态获取位置信息，其代码为：

```
package fs.gps_test;
import android.app.Activity;
import android.content.Context;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.widget.EditText;
public class MainActivity extends Activity {
    //定义 LocationManager 对象
    private LocationManager locationManager;
    private EditText show;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.main);
        show = (EditText) findViewById(R.id.main_et_show);
        //获取系统 LocationManager 服务
        locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
        //从 GPS 获取最近的定位信息
        Location location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
        //将 location 里的位置信息显示在 EditText 中
        updateView(location);
        //设置每 2 秒获取一次 GPS 的定位信息
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,2000, 8, new LocationListener() {
                    @Override
                    public void onLocationChanged(Location location) {
                        //当 GPS 定位信息发生改变时,更新位置
                        updateView(location);
                    }
                    @Override
                    public void onProviderDisabled(String provider) {
                        updateView(null);
                    }
                    @Override
                    public void onProviderEnabled(String provider) {
                        //当 GPS LocationProvider 可用时,更新位置
                        updateView(locationManager
                                .getLastKnownLocation(provider));
                    }
                    @Override
                    public void onStatusChanged(String provider, int status,Bundle extras) {
                    }
                });
    }
    private void updateView(Location location) {
        if (location!= null) {
            StringBuffer sb = new StringBuffer();
            sb.append("实时的位置信息: \n 经度: ");
            sb.append(location.getLongitude());
            sb.append("\n 纬度: ");
            sb.append(location.getLatitude());
            sb.append("\n 高度: ");
            sb.append(location.getAltitude());
            sb.append("\n 速度: ");
            sb.append(location.getSpeed());
            sb.append("\n 方向: ");
            sb.append(location.getBearing());
            sb.append("\n 精度: ");
            sb.append(location.getAccuracy());
            show.setText(sb.toString());
        } else {
            //如果传入的 Location 对象为空,则清空 EditText
            show.setText("");
        }
    }
}
```

(4) 打开 AndroidManifest.xml 文件,用于设置获取位置信息权限,其代码为:

```
⋮
    </application>
    <!-- 设置 GPS 定位权限 -->
```

```
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
</manifest>
```

运行程序，并返回到 DDMS 的 Emulator Control 面板中，单击面板中的 send 按钮，即可向 Android 模拟器发送 GPS 定位信息，获得 GPS 位置信息如图 10-20 所示。



图 10-20　GPS 动态获取位置信息

网上参考资源

http://baike. baidu. com/link?url=tDsJMmSQQcc7MDrN69NVPFvZZx7AZu8v_W1CCIf7sbPOgYRKtK8U6My1iqbmpQEV7AY16dcyp0Fxz8yRdH4qEa
http://blog. csdn. net/android_tutor/article/details/7645486
http://www. cnblogs. com/menlsh/archive/2012/11/18/2776003. html
http://www. cnblogs. com/dyllove98/p/3202887. html
http://www. cnblogs. com/greatverve/archive/2012/03/15/android-carema. html
http://www. jb51. net/article/40880. htm
http://blog. csdn. net/a497393102/article/details/8593406
http://www. apkbus. com/android-113949-1-1. html
http://maimode. iteye. com/blog/1673384
http://blog. csdn. net/zengshuqin/article/details/7099499
http://www. 2cto. com/kf/201310/249938. html
http://blog. 163. com/langfei520@yeah/blog/static/1727102222011111004662/
http://blog. csdn. net/xlcs2012/article/details/8050465
http://www. cnblogs. com/MyJie/archive/2012/01/20/2328078. html
http://blog. csdn. net/sxsj333/article/details/6368246
http://blog. csdn. net/wangkuifeng0118/article/details/7339578
http://www. linuxidc. com/Linux/2014-02/96397. htm
http://www. cnblogs. com/hanyonglu/archive/2012/03/26/2418178. html
http://wenku. baidu. com/link? url = WMmXmQLgK3IFa7mkvtZdBS62TYWxK9S1fOBB4iDns_LRtBRdz70tqtvHi52maz2LZ2D4ZjVffx_mRRjdcP3svPnJMd1ciErsd3Ixs7HK_OS
http://www. cnblogs. com/menglin2010/archive/2011/12/27/2303214. html
http://lszdb1983. blog. 163. com/blog/static/20426348201173043857757/
http://www. jb51. net/article/38641. htm
http://www. cnblogs. com/hanyonglu/archive/2012/02/19/2357842. html
http://52android. blog. 51cto. com/2554429/496621
http://blog. csdn. net/w7849516230/article/details/7074446
http://blog. csdn. net/zlqqhs/article/details/8759350
http://wenku. baidu. com/link? url = QwgdI782MVdT7suh_5KgQ-QdI1YM8toFGlNXFtv5E671rq1-WGlsl4T1EZ9f_oBp-5Cw61w93glFfj4at0rbv2n79Bk8Fk1lAws1hIBCGvq
http://nannan408. iteye. com/blog/1905955
http://www. cnblogs. com/gzggyy/archive/2011/06/16/2082809. html
http://blog. csdn. net/a859522265/article/details/7852868
http://blog. csdn. net/cjjky/article/details/6881582
http://blog. csdn. net/meizhen51/article/details/6155912
http://www. cnblogs. com/linjiqin/archive/2011/03/10/1980215. html
http://aina-hk55hk. iteye. com/blog/679134/
http://www. 2cto. com/kf/201205/131876. html
http://bbs. 51cto. com/thread-1009791-1. html
http://blog. csdn. net/dadahacker/article/details/5742174
http://www. cnblogs. com/menglin2010/archive/2012/03/03/2376526. html

http://www.oschina.net/code/snippet_54100_1422
http://blog.csdn.net/lee576/article/details/7900228
http://www.jcodecraeer.com/a/anzhuokaifa/2012/0818/358.html
http://www.cnblogs.com/bavariama/archive/2013/04/22/3036068.html
http://www.cnblogs.com/wolipengbo/p/3387774.html
http://blog.csdn.net/aminfo/article/details/7847761
http://www.cnblogs.com/happyDays/archive/2013/08/01/3230980.html
http://www.jb51.net/article/32341.htm
http://www.360doc.com/content/13/0102/22/6541311_257754535.shtml
http://wjlgryx.iteye.com/blog/1124029
http://www.haogongju.net/art/1472860
http://www.eoeandroid.com/forum.php?mod=viewthread&tid=564
http://www.apkbus.com/android-14231-1-1.html
http://www.cnblogs.com/salam/archive/2010/10/19/1855511.html
http://www.cnblogs.com/salam/archive/2010/10/06/1844660.html
http://www.cnblogs.com/salam/archive/2010/10/07/1845089.html
http://wenku.baidu.com/link?url=_qxPh7U9RZAY2dHdvAUTfIc7ATlnLr7eGZI8EhJYNuD_MpC5rVJLOFn24Fj-wT_RZWHtIZYctXj8inUXLtZr8n2kE-eLLpuhScRnfsblwMO
http://wenku.baidu.com/browse/downloadrec?doc_id=e2ccb36ba45177232f60a2ef&
http://www.2cto.com/kf/201110/106615.html
http://www.cnblogs.com/linjiqin/archive/2011/02/21/1960185.html
http://android.tgbus.com/Android/tutorial/201107/360548.shtml
http://www.jizhuomi.com/android/example/134.html
http://www.cnblogs.com/salam/archive/2010/10/06/1844741.html
http://my.oschina.net/zhoulc/blog/127065
http://hi.baidu.com/wustrive_2008/item/8430ad58cbeae90ce7c4a565
http://www.eoeandroid.com/thread-66783-1-1.html
http://android.tgbus.com/Android/tutorial/201105/351259.shtml
http://blog.csdn.net/x605940745/article/details/9703845
http://blog.csdn.net/x605940745/article/details/9704229
http://blog.csdn.net/sun6255028/article/details/6685075
http://byandby.iteye.com/blog/815212
http://www.cnblogs.com/mengdd/archive/2012/12/20/2826235.html
http://www.blogjava.net/zhip/archive/2011/01/25/343511.html
http://wenku.baidu.com/browse/downloadrec?doc_id=8f5d0fc46137ee06eff91851&
http://baike.baidu.com/subview/1241829/9322617.htm?fr=aladdin
http://tieba.baidu.com/p/2359113878
http://www.oschina.net/question/16_22930
http://blog.chinaunix.net/uid-26524139-id-3143948.html
http://www.2cto.com/kf/201109/105226.html
http://blog.csdn.net/android_jiangjun/article/details/25346627
http://blog.csdn.net/pku_android/article/details/7343258
http://blog.csdn.net/qinde025/article/details/6833888
http://imshare.iteye.com/blog/771539
http://www.cnblogs.com/linjiqin/archive/2011/02/22/1960890.html

参 考 文 献

[1] 李刚. 疯狂 Android 讲义. 北京：电子工业出版社. 2012.
[2] 张余. Android 网络开发从入门到精通. 北京：清华大学出版社. 2014.
[3] 欧阳零. Android 编程兵书. 北京：电子工业出版社. 2014.
[4] 李佐彬. Android 开发入门与实战体验. 北京：机械工业出版社. 2011.
[5] 软件开发技术联盟. Android 开发实战. 北京：清华大学出版社. 2013.
[6] 李刚. 疯狂 Android 讲义. 北京：电子工业出版社. 2011.
[7] 楚无咎. Android 编程经典 200 例. 北京：电子工业出版社. 2013.
[8] 高洪岩. Android 学习精要. 北京：清华大学出版社. 2012.
[9] 明日科技. Android 从入门到精度. 北京：清华大学出版社. 2012.
[10] 李波，史江萍，王祥凤. Google Android 4.X 从入门到精通. 北京：清华大学出版社. 2012.